ImageTTR: Grounding Type Theory with Records in Image Classification for Visual Question Answering

Arild Matsson
Språkbanken, Department of Swedish
University of Gothenburg
arild.matsson@gu.se

Simon Dobnik **Staffan Larsson**
CLASP and FLoV
University of Gothenburg
simon.dobnik@gu.se
sl@ling.gu.se

Abstract

We present ImageTTR, an extension to the Python implementation of Type Theory with Records (pyTTR) which connects formal record type representation with image classifiers implemented as deep neural networks. The Type Theory with Records framework serves as a knowledge representation system for natural language the representations of which are grounded in perceptual information of neural networks. We demonstrate the benefits of this symbolic and data-driven hybrid approach on the task of visual question answering.

1 Introduction

A situated artificial conversational agent must be able to interact with its environment through perception and action, as well as with other agents through language. The key challenge for building situated agents is how to represent and reason with both linguistic and non-linguistic information in some formal system that can be modelled on a computer (the question of information fusion). This is challenging because linguistic and perceptual domains are not of the same kind. Perceptional information is typically represented by abundance of sensory readings that are typically represented as real numbers. On the other hand, language is a symbolic system. The two domains are typically bridged through classification (Harnad, 1990).

Several approaches have dealt with information fusion by combining machine learning classification with some kind of formal representation (Kruijff et al., 2006, 2007; Roy, 2005). Recently, using deep learning architectures such associations are learned automatically (Xu et al., 2015; Lu et al., 2016). While they are highly effective, they are not interpretable and directly portable to domains where one would like to combine them with logic-like formal representations.

Among formal representations, first-order logic (FOL) have long been of choice, but it has its limitations. A recent contribution that combines several branches in formal systems is Type Theory with Records (TTR) (Cooper, 2005b, fthc).

The goal of this paper is a model, which is formulated in TTR and connects machine-learning classification of perceptual information on one hand with language on the other, as well as an executable implementation of this model. TTR is expected to represent semantics of perceptual and linguistic content in a single framework. The connection is realised in a visual question answering (VQA) application, where questions are interpreted in the context of an image.

2 Background

2.1 Type Theory with Records (TTR)

Type Theory with Records (TTR) is a formal semantic framework that represents the semantics of language, action and perception (Cooper, fthc). The *types* in type theory are employed to represent words

Proceedings of the IWCS 2019 Workshop on Computing Semantics with Types, Frames and Related Structures, pages 55–64
Gothenburg, Sweden, May 24, 2019.

and semantic units within a sentence or discourse (Cooper, 2005b). A brief introduction to TTR is given in this section. For further reference, see Cooper (2005a) and Cooper (2012).

To begin with, $a : T$ is the judgement that a is of type T. Some *basic types* in TTR are Ind, the type of an individual, and Int, the type of integers.

Given that T_1 and T_2 are types, $T_1 \rightarrow T_2$ is a *functional type* whose domain is objects of type T_1 and whose range is objects of type T_2.

Next, we introduce *records* and *record types*. If $a_1 : T_1, a_2 : T_2(a_1), \ldots, a_n : T_n(a_1, a_2, \ldots, a_{n-1})$, where $T(a_1, \ldots, a_n)$ represents a type T which depends on the objects $a_1, \ldots, a_n$, the record to the left in Equation 1 is of the record type to the right. In Equation 1, $\ell_1, \ldots, \ell_n$ are *labels* which can be used elsewhere to refer to the values associated with them. A sample record and record type is shown in Equation 2. If r is a record and ℓ is a label in r, we can use a *path* $r.\ell$ to refer to the value of ℓ in r. Similarly, if T is a record type and ℓ is a label in T, $T.\ell$ refers to the type of ℓ in T. Records (and record types) can be nested, so that the value of a label is itself a record (or record type).

$$r = \begin{bmatrix} \ell_1 = a_1 \\ \ell_2 = a_2 \\ \ldots \\ \ell_n = a_n \\ \ldots \end{bmatrix} : \begin{bmatrix} \ell_1 : T_1 \\ \ell_2 : T_2(\ell_1) \\ \ldots \\ \ell_n : T_n(\ell_1, \ell_2, \ldots, \ell_{n-1}) \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} \text{ref} = obj_{123} \\ \text{c}_{\text{man}} = \text{prf}(\text{man}(obj_{123})) \\ \text{c}_{\text{run}} = \text{prf}(\text{run}(obj_{123})) \end{bmatrix} : \begin{bmatrix} \text{ref} : Ind \\ \text{c}_{\text{man}} : \text{man}(\text{ref}) \\ \text{c}_{\text{run}} : \text{run}(\text{ref}) \end{bmatrix} \tag{2}$$

As can be seen in Equation 2, types can be constructed from predicates, e.g., "run" or "man". Such types are called *ptypes* and correspond roughly to predicate formulas in first order logic. Ptypes may be *dependent*, which is represented by the fact that arguments to the predicate may be represented by labels used elsewhere. In Equation 2, the type of c_{man} is dependent on ref (as is the type of c_{run}).

A fundamental type-theoretical intuition is that something of a ptype $P(a_1, \ldots, a_n)$ is whatever it is that counts as a proof of $P(a_1, \ldots, a_n)$. One way of putting this is that "propositions are types of proofs". In Equation 2, we simply use prf(P) as a placeholder for proofs of P; below, we will show how perceptual input can be included in proofs[1].

A *singleton type* T_a is a subtype of T restricted so that only a can be a witness of it. In a record type, a singleton-typed field can be written as a *manifest field*: $\left[\text{ x } : \ T_a \ \right] = \left[\text{ x} = a \ : \ T \ \right]$

Types are sorted into orders, where types of one order may be witnesses of a type in a higher order. $Type^n, n > 0$ is the type of all types of order $n - 1$. In this paper, most types will be of order 0, so we will skip the order superscript and use $Type$ to denote $Type^1$. Similar to $Type$, $RecType$ is the type of all record types and $PType$ is the type of all ptypes.

The *re-labelling* η of a record type T is a set of tuples where the first element is a label in T and the second is another, new label. T_η is the record type which has the same fields as T but where the first item in each element of η has been replaced with the second item. Thus, if $T = \left[\text{ x } : \ T' \ \right]$ and $\eta = [\langle \text{x, y} \rangle]$, then $T_\eta = \left[\text{ y } : \ T' \ \right]$.

A list of objects of type T is a witness of the *list type* list(T): If L is a list and $\forall a \in L, a : T$, then $L : \text{list}(T)$. In this paper, the list containing a, b and c will be written as $[a, b, c]$.

2.2 Visual Question Answering (VQA)

Antol et al. (2017) suggest visual question answering (VQA) as a challenge for multi-modal semantic systems. A VQA system is presented an image and a natural-language question about the image, and is expected to produce a natural-language answer. The initiative includes datasets and a series of annual competitions since 2016.

[1] Note that TTR is not proof-theoretic like may other type theories. TTR proofs are more like *witnesses* in situation semantics (Barwise and Perry, 1983) or the *proof objects* in intuitionistic type theory (Martin-Löf and Sambin, 1984).

This task has been defined within the deep learning approach to vision and language. However, a purely end-to-end approach faces challenges like opacity and ignorance of developed models of perception. We choose a hybrid approach to VQA and provide a proof of concept of its strengths in this domain.

Many VQA problems are formulated with open-domain questions. In this paper, however, we are limited to polar (*yes/no*) questions, because those are, arguably, the easiest to model and answer.

It is important to note that the presented model is not an attempt at scoring high in the VQA task. A comparison to dedicated VQA models would place this model low in most aspects: limitations in language domain and syntax, and computation speed, just to name a couple. Rather, this is an exploration of a multimodal representation model (or paradigm), with VQA used as an evaluation context. Advantages of the model include modularity, transparency and reversibility, as provided by the underlying formal-semantic framework.

2.3 Tools

PyTTR (Cooper, 2017) is a Python implementation of TTR. It supports the modelling of TTR types and operations such as judgement and type checking. As a Python library it also enables other features and peripheral procedures to be written in Python. PyTTR allows, in turn, the implementation of TTR models. By implementing a theoretical model as a computer program, it can "come alive" and be tested on real problems and data. When implemented, the model can be evaluated in practical settings.

You Only Look Once (YOLO) (Redmon et al., 2016) is a neural network model for image recognition. Given an image, it will detect objects and classify them. Each detection consists of a bounding box in pixel coordinates, a class label and a confidence score. YOLO is written in C, using the Darknet neural network library (Redmon, 2013). It can be used in Python with the Darkflow library (Trieu, 2018). Development within this thesis has been using the YOLOv2 configuration (Redmon, 2018) trained on the COCO dataset (Lin et al., 2014).

Natural Language Toolkit (NLTK) (Bird et al., 2009) is a Python library facilitating various natural language processing operations. It features a semantically augmented context-free grammar (CFG) framework which enables parsing language into first-order logic (FOL) formulas.

3 A grounded PyTTR

The Python implementation of the model described here is published as a Jupyter notebook file at `https://github.com/arildm/imagettr/releases/tag/1.1` under the open-source MIT license. This section contains references to numbered cells of the notebook file.

3.1 Object detection

The perception of objects in this model is largely based on (Dobnik and Cooper, 2017, Section 5.1). The name and definitions of some of the types have been slightly modified here, for better alignment with the names used in the implementation within this project.

Let the world be of some type $World$, and any portion of the world of some type $Segment$. These types are left undefined for now, as they differ significantly between (Dobnik and Cooper, 2017) and this project. An *object detector* function $f_{objdetector} : ObjDetector$ (Equation 3) maps the world to a collection of *perceptual objects* of the type Obj (Equation 4). A perceptual object contains a segment of the world as well as a field of the type $Ppty$ (Equation 5). A $Ppty$ function can be applied to an individual and return a type, for example $\lambda v : Ind$. kite$(v) : Ppty$. The type $PType$ is defined as the type of all ptypes (Definition 1).

Definition 1 *For any ptype* $T = pred(v_1, ..., v_n)$, $T \sqsubseteq PType$.

$$ObjDetector = (World \rightarrow \text{list}(Obj)) \tag{3}$$

$$Obj = \begin{bmatrix} \text{seg} & : & Segment \\ \text{pfun} & : & Ppty \end{bmatrix} \tag{4}$$

$$Ppty = (Ind \rightarrow PType) \tag{5}$$

The perceptual object is evidence that a certain segment of the perceptual input is associated with a certain property (such as being a kite). Going further, an *individuation function* $f_{indfun} : IndFun$ (Equation 6) generates an *individuated object* from each perceptual object. The individuated object is a record type and a subtype of $IndObj$ (Equation 7). Here, the 'x' field refers to a specific individual which was only implied by the existence of the perceptual object. The 'cl' field specifies that the position of 'x' is the content of the field 'loc', which has the same type as 'seg' in Obj. Through the 'cp' field, the property can be explicitly associated with the individual.

$$IndFun = (Obj \rightarrow RecType) \tag{6}$$

$$IndObj = \begin{bmatrix} \text{x} & : & Ind \\ \text{loc} & : & Segment \\ \text{cp} & : & PType \\ \text{cl} & : & \text{location}(\text{x}, \text{loc}) \end{bmatrix} \tag{7}$$

Note that the step from the perceptual to the *conceptual* domain is made by generating a record type (not a record), namely the type of *situations* where a certain individual is at a certain location.

3.1.1 Model

This section applies the theory outlined above to the case at hand. The main difference against (Dobnik and Cooper, 2017) is that the world is now an image (Equation 8), rather than a 3D point space. A $Segment$ (Equation 9) is now defined as a record type describing a rectangular bounding box within an image. Its fields contain the centre coordinates of the box ('cx' and 'cy') and the width ('w') and height ('h') of the box.

$$World = Image \tag{8}$$

$$Segment = \begin{bmatrix} \text{cx} & : & Int \\ \text{cy} & : & Int \\ \text{w} & : & Int \\ \text{h} & : & Int \end{bmatrix} \tag{9}$$

The object detection function is a Python function which takes an image as input and invokes YOLO (the implementation of this procedure is in the notebook cells 11, 12 and 14). The return value from YOLO is a collection of dictionary objects, which are converted to perceptual objects of the type Obj which are the output of the object detection function (cell 14, see example in Equation 11).

The individuation function f_{indfun} is defined in Equation 10 (cell 15). Here, the 'x' field is specified as a newly instantiated Ind object a_n, and 'loc' is specified as the perceptual object's 'seg'. The type of the 'cp' field applies the perceptual object's 'pfun' to 'x'. An example output from the function is shown in Equation 12.

$$f_{indfun} = \lambda r : Obj \,.\, \begin{bmatrix} \text{x} = a_n & : & Ind \\ \text{cp} & : & r.\text{pfun}(\text{x}) \\ \text{cl} & : & \text{location}(\text{x}, \text{loc}) \\ \text{loc} = r.\text{seg} & : & Segment \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} \text{seg} = \begin{bmatrix} \text{cx} = 102 \\ \text{cy} = 156 \\ \text{w} = 204 \\ \text{h} = 84 \end{bmatrix} \\ \text{pfun} = \lambda v : Ind \,.\, \text{kite}(v) \end{bmatrix} : Obj \tag{11}$$

$$\begin{bmatrix} \text{x} = a_0 & : & Ind \\ \text{cp} & : & \text{kite}(\text{x}) \\ \text{cl} & : & \text{location}(\text{x}, \text{loc}) \\ \text{loc} = \begin{bmatrix} \text{cx} = 102 \\ \text{cy} = 156 \\ \text{w} = 204 \\ \text{h} = 84 \end{bmatrix} & : & Segment \end{bmatrix} \sqsubseteq IndObj \tag{12}$$

3.2 Spatial relations

(Dobnik and Cooper, 2013, Section 3) provides a TTR model of the classification of spatial relations between a located object, a reference object and a viewpoint. The classifier for a given spatial relation may be equal to a geometric classifier κ or this can be combined with functional classifiers to encompass functional aspects of objects on spatial relations (Coventry et al., 2001).

In this project, spatial relation classification is less sophisticated, ignoring the viewpoint as well as the functional aspects of a spatial relation. The reference frame is implicit in the frame of an image, rather than given by a viewpoint object.

A tuple-like record of the type $LocTup$ (Equation 13) groups a located object 'lo' and a reference object 'refo'. A classifier of the type $RelClf$ (Equation 14) takes a $LocTup$ record and returns a new record type which describes the relation.

$$LocTup = \begin{bmatrix} \text{lo} & : & IndObj \\ \text{refo} & : & IndObj \end{bmatrix} \tag{13}$$

$$RelClf = (LocTup \rightarrow RecType) \tag{14}$$

A pattern for a $RelClf$ classifier is given in Equation 15, where rel is to be replaced with a predicate and κ_{rel} with a boolean classifier.

$$\lambda r : LocTup \; . \begin{cases} \begin{bmatrix} \text{x} & : & r.\text{lo.x} \\ \text{y} & : & r.\text{refo.x} \\ \text{cr} & : & rel(\text{x}, \text{y}) \end{bmatrix}, & \text{if } \kappa_{rel}(r.\text{lo.loc}, r.\text{refo.loc}) \\ [\,], & \text{otherwise} \end{cases} \tag{15}$$

The boolean classifiers are implemented as Python functions, one for each of the relations 'left', 'right', 'above' and 'below'. Each returns true or false after comparing the 'cx' or 'cy' fields of the two $Segment$ inputs, i.e. the centre points of the bounding boxes of the objects.

The whole procedure described in this section is implemented in notebook cell 16.

3.3 Beliefs

The set of individuated objects and the set of relation classification results form a set of beliefs. They contain information that the agent has grounded about its perceptual environment. Each of these types is a situation held to be true, by virtue of resulting from perception mechanisms. The belief types are *combined* into one scene record type S which describes the full scene.

3.3.1 Combining situation types

In TTR literature, the combination of multiple record types into one typically follows one of two methods. The first method uses the merge operation $\wedge$ or the asymmetric merge operation $\boxed{\wedge}$. The reliance on field labels in the (asymmetric) merge operation is a problem in this case, where labels have been automatically generated and sometimes clash. Another method is iteratively nesting one record type as a field of the next, and then flattening the result to avoid the nesting. This method avoids label clashes.

However, a third method is used in this project for the purpose of computational speed (cell 18). Each belief record type is re-labelled to only have unique labels, and they are then merged. The resulting type is essentially the same as in the case when nesting and flattening.

3.3.2 De-duplication

Among the belief record types, the same individuals occur more than once. For instance, one belief may hold that a_1 is a kite and another that a_1 is above some other individual. Both beliefs will have a field like $\text{x}_i = a_1 : Ind$, with different labels x_i but the same specification to a_1. *De-duplicating* these is necessary for the subtype check that will follow. This process involves first finding which fields have the same type as another field. Subsequently, simply removing duplicates is not an option, as there may be other fields that depend on the duplicate field. These dependent fields must also first be updated to use the remaining field. This algorithm is implemented in cell 7.

3.4 Parsing user questions

The VQA setting requires the model to understand not only the visual input, but also a natural language question. Parsing natural language is a complex task, ambiguity in syntax as well as semantics being one significant source of difficulties. Within this project, this task has been drastically reduced by focusing on a tiny language domain, with only a handful grammatical constructions and a small, customised vocabulary.

Theoretical formulations of syntactic parsing to TTR have been given in Cooper (2005a,b, 2012, fthc). Applying them in this project is however considered out of scope. Instead, we make use of parsing tools from NLTK (Bird et al., 2009), as follows (cell 20).

A small context-free grammar (CFG) is composed and used to parse natural language into a representation of first-order logic (FOL). The parsing process is visualised in Figure 1. The FOL is then "translated" into a TTR representation by traversing the FOL expression tree and gradually building a TTR record type, according to the following rules: For an *Exists* expression, an Ind field is added to the type. For an *Application* expression, a ptype field is added, copying the predicate and variable names. An *And* expression simply triggers recursion into each of the two terms.

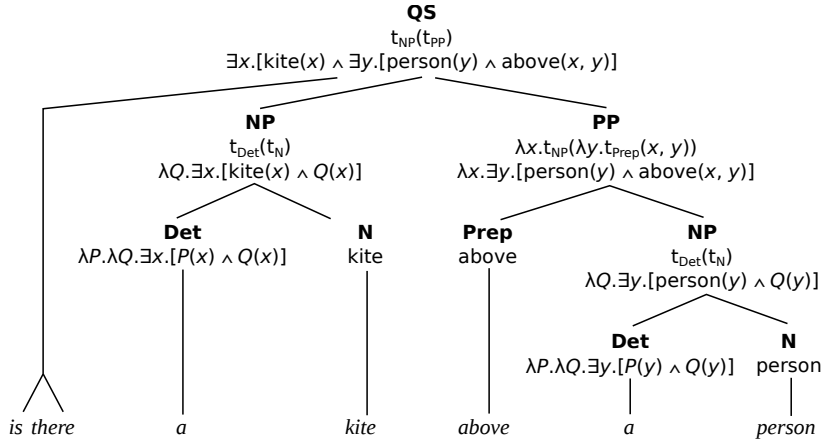

Figure 1: Syntactic-semantic parsing of an utterance into first-order logic. Each node in the tree has a bold-faced constituent label and the FOL lambda expression associated with it. Parent constituents additionally have a third line containing the formula resulting from substitution and β-reduction.

The question sentence in Figure 1 is now translated to the record type Q in Equation 16.

$$Q = \begin{bmatrix} \text{x} & : & Ind \\ \text{y} & : & Ind \\ \text{c}_0 & : & \text{kite}(\text{x}) \\ \text{c}_1 & : & \text{person}(\text{y}) \\ \text{c}_2 & : & \text{above}(\text{x}, \text{y}) \end{bmatrix} \tag{16}$$

3.5 Question answering

As a limitation of scope, this project focuses on polar questions. Thus, the language is first limited in domain, to various nouns and geometric spatial relations, and then grammatically, to polar questions.

Object detection and spatial relation classification result in a collection of situation types, which are combined to one, the *scene* type S (Section 3.3). Aside from that, the language parsing results in a type representing the *question* Q (Section 3.4).

The scene type is considered true by virtue of being generated by mechanisms of perceptual classification. The situation described by the question type, on the other hand, will be true if there exists a witness of that type, $r : Q$. It follows that the question type is true if it is a super-type of the scene type, $S \sqsubseteq Q$. Thus, rather than looking for a witness to the question type, we formulate the problem as subtype checking, and answer the question with "yes" or "no" depending on the truth of that check.

An important problem, however, stems from the fact that TTR record types are labelled. In general, fields in the scene type and question type will not share labels in a way that enables direct subtype checking to be useful. Field labels in the scene type will generally not agree with those in the question type. The remedy to this is an alternative subtype relation $\sqsubseteq_{\text{rlb}}$ which is insensitive to label names (cell 21).

Definition 2 *A record type S is a **re-label-subtype** of the record type Q, $S \sqsubseteq_{rlb} Q$, if there is a re-labelling η of Q such that $S \sqsubseteq Q_\eta$.*

An intuitive way to implement this would be to perform all re-labellings η from labels of Q to labels of S and check whether the subtype relation holds. However, this approach is practically impossible, as the number quickly grows very large. An alternative algorithm is developed for the purpose of this project, where fast computation is enabled by making a few assumptions about the input record types[2].

The algorithm handles non-dependent and dependent fields separately. First, when considering re-labellings η of Q, only the non-dependent fields are included. This drastically limits the number of re-labellings to try. Then, for each re-labelling $Q_{\eta'}$ being tried, the remaining, dependent fields are subtype-checked individually, in order to avoid more re-labelling. This means checking $\text{person(x)} \sqsubseteq \text{person(x)}$ (true) instead of $\left[\ \text{c} \ : \ \text{person(x)}\ \right] \sqsubseteq \left[\ \text{f} \ : \ \text{person(x)}\ \right]$ (false). For each dependent field in $Q_{\eta'}$ where there is a field in S that passes this check, the corresponding label pair is added to η'. If, for some $Q_{\eta'}$ field, there is no matching S field, it is concluded that $S \not\sqsubseteq Q_{\eta'}$, and the algorithm proceeds to the next η.

When a re-labelling η is found which enables the subtype check for all dependent fields to pass, the algorithm returns η, which can be interpreted as $S \sqsubseteq_{\text{rlb}} Q$ being true. If all non-dependent-field re-labellings have been evaluated without successful dependent-field subtype checks, the algorithm returns nothing, which is interpreted as the relation not holding.

4 The agent structure

The perceptual-conceptual pieces described above are now connected in an *agent* record type (Equation 17 and Equation 18, cell 21) with associated manipulation algorithms (cell 25). Upon receiving an image, it will carry out object detection, individuation and spatial relation classification, in order to form its beliefs. It may also receive a parsed natural-language utterance, which will then be verified against the beliefs. A construction like this provides a means to answer natural-language questions about the image.

$$Agent = \begin{bmatrix} \text{objdetector} & : & ObjDetector \\ \text{indfun} & : & IndFun \\ \text{relclfs} & : & \text{list}(RelClf) \\ \text{state} & : & AgentState \end{bmatrix} \quad (17)$$

$$AgentState = \begin{bmatrix} \text{img} & : & Image \\ \text{perc} & : & \text{list}(Obj) \\ \text{bel} & : & \text{list}(RecType) \\ \text{utt} & : & String \\ \text{que} & : & RecType \end{bmatrix} \quad (18)$$

The fields 'objdetector', 'indfun' and 'relclfs' of $Agent$ are to be statically defined for a specific agent. The $AgentState$ record in 'state' will be modified by the agent algorithm while running. The 'perc' field will contain a list of perceptual objects. The 'bel' field will be a list of beliefs modelled as situation types: individuated objects and spatial relations between individuals.

For an agent record $ag : Agent$, the perception and question-answering procedure is carried out as follows. Visual input in the form of an image is received, and object detection returns a collection of perceptual objects, for which individuated objects are generated and added as beliefs (Section 3.1). For each pair of individuated objects, spatial relation classifications are generated and also added to beliefs (Section 3.2). Natural-language input is parsed to a type representing the situation hypothesised in the

[2]The algorithm presupposes a certain conformity between S and Q, in that it is not aware of the equivalence between a record type and its flattened version. For instance, it will fail to acknowledge $\left[\ x_1 \ : \ \left[\ x_2 \ : \ T\ \right]\ \right] \sqsubseteq_{\text{rlb}} \left[\ x_3 \ : \ T\ \right]$. However, this problem is not encountered in this application, due to the the way that S and Q are constructed.

question (Section 3.4). Finally, a re-label-subtype check is performed against the beliefs, and the answer "Yes" or "No" is emitted (Section 3.5).

5 Discussion

Within this project, the foundations of visual question answering (VQA) have been implemented in Type Theory with Records (TTR). The result is an executable application powered by PyTTR.

The application is a practical example that TTR can be used to connect existing vision and language systems. TTR is the single framework that serves as a knowledge representation system that glues together the various parts of the pipeline – perception, language and grounding – and provides reasoning qualified for a simple VQA application.

PyTTR This is one of the few applications of the recently developed PyTTR library. A few extensions were needed in order to realise the present project. Simple and general operations, like copying a record type, could be implemented directly in the PyTTR library. Others, like the combination of record types (Section 3.3.1), should remain in the project-specific source code.

Inference The use of formal frameworks for question-answering tasks, as opposed to statistical or neural methods, especially invites techniques for logical inference. Consider an image of a person wearing glasses, and the question "Does this person have 20/20 vision?" It is reasonable to assume that a person is wearing glasses because they do not have perfect eyesight, to which "20/20 vision" is synonymous. Logical inference, in connection with a database of real-world knowledge encoded in TTR, could help to achieve the synonymity as well as the relationship between eyesight and wearing glasses.

Theorem prover approach to subtype check The re-label-subtype $\sqsubseteq_{\text{rlb}}$ check, currently implemented as an iterative algorithm, could likely be made more efficient and generalised if instead cast as a problem of theorem proving (Plaisted, 2014).

5.1 Future work

Non-polar questions Extending the language domain should be an interesting topic for further research. Keeping within the problem domain of geometric spatial relations, allowing other question types than polar questions is one direction to explore. (Dobnik, 2009, p. 156) lists four basic question types: "Where is the chair?", "Is the table to the left of the chair?" (this is the focus of this project), "What is to the left of the chair?" and "What is the chair to the left of?"

Inference first The algorithm of perception performs classification of spatial relations on all pairs of individuated objects. In other words, all of the agent's beliefs are inferred at once. Later, when attempting to answer the given question, the beliefs can be queried directly in the subtype check. This means spending more effort than sometimes necessary. A more viable alternative is to first parse the question and then perform inference as needed to arrive to an answer. If the question is about the spatial relation between a kite and a person, it will probably be enough to find a kite and a person in the scene, and check that the spatial relation between them matches the one expressed in the question.

Sophisticated spatial relation classification The implementation of spatial relation classification in this project compares the horizontal or vertical coordinates of the centre points of two objects, a simplified geometric representation that does not correspond well to human judgements. A more sophisticated geometric method is the statistical *spatial templates* model (Logan and Sadler, 1996). Another is the *attentional vector-sum (AVS)* model, a mathematical formula which respects object shape (Regier and Carlson, 2001).

References

Antol, S., A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh (2017, May). VQA: Visual Question Answering. *International Journal of Computer Vision 123*(1), 4–31.

Barwise, J. and J. Perry (1983). *Situations and Attitudes*. The MIT Press.

Bird, S., E. Klein, and E. Loper (2009). *Natural Language Processing with Python* (1st ed.). O'Reilly Media, Inc.

Cooper, R. (2005a). Austinian truth, attitudes and type theory. *Research on Language and Computation 3*, 333–362.

Cooper, R. (2005b, April). Records and Record Types in Semantic Theory. *Journal of Logic and Computation 15*(2), 99–112.

Cooper, R. (2012). Type theory and semantics in flux. In R. Kempson, N. Asher, and T. Fernando (Eds.), *Handbook of the Philosophy of Science*, Volume 14: Philosophy of Linguistics. Elsevier BV. General editors: Dov M. Gabbay, Paul Thagard and John Woods.

Cooper, R. (2017). PyTTR. `https://github.com/GU-CLASP/pyttr`.

Cooper, R. (fthc). Type Theory and Language: From Perception to Linguistic Communication. Draft of book chapters available from `https://sites.google.com/site/typetheorywithrecords/drafts` (accessed on 2018-01-17), University of Gothenburg.

Coventry, K., M. Prat-Sala, and L. Richards (2001, April). The Interplay between Geometry and Function in the Comprehension of Over, Under, Above, and Below. *Journal of Memory and Language 44*, 376–398.

Dobnik, S. (2009). *Teaching Mobile Robots to Use Spatial Words*. Ph. D. thesis, The Queen's College, University of Oxford.

Dobnik, S. and R. Cooper (2013, March). Spatial Descriptions in Type Theory with Records. In *Proceedings of IWCS 2013 Workshop on Computational Models of Spatial Language Interpretation and Generation (CoSLI-3)*, Potsdam, Germany, pp. 1–6. Association for Computational Linguistics.

Dobnik, S. and R. Cooper (2017). Interfacing Language, Spatial Perception and Cognition in Type Theory with Records. *Journal of Language Modelling 5*(2), 273–301.

Harnad, S. (1990, June). The symbol grounding problem. *Physica D 42*(1–3), 335–346.

Kruijff, G.-J. M., J. D. Kelleher, and N. Hawes (2006). Information fusion for visual reference resolution in dynamic situated dialogue. In E. André, L. Dybkjær, W. Minker, H. Neumann, and M. Weber (Eds.), *Perception and Interactive Technologies. International Tutorial and Research Workshop, PIT 2006 Kloster Irsee, Germany*, pp. 117–128. Berlin, Heidelberg: Springer.

Kruijff, G.-J. M., H. Zender, P. Jensfelt, and H. I. Christensen (2007). Situated dialogue and spatial organization: what, where... and why? *International Journal of Advanced Robotic Systems 4*(1), 125–138.

Lin, T.-Y., M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick (2014). Microsoft COCO: Common Objects in Context. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014*, Volume 8693, pp. 740–755. Cham: Springer International Publishing.

Logan, G. D. and D. D. Sadler (1996). A computational analysis of the apprehension of spatial relations. In *Language and Space.*, Language, speech, and communication., pp. 493–529. Cambridge, MA, US: The MIT Press.

Lu, J., C. Xiong, D. Parikh, and R. Socher (2016). Knowing when to look: Adaptive attention via a visual sentinel for image captioning. *arXiv arXiv:1612.01887 [cs.CV]*, 1–10.

Martin-Löf, P. and G. Sambin (1984). *Intuitionistic type theory*. Studies in proof theory. Bibliopolis.

Plaisted, D. A. (2014, March). Automated theorem proving: Automated theorem proving. *Wiley Interdisciplinary Reviews: Cognitive Science 5*(2), 115–128.

Redmon, J. (2013). Darknet: Open Source Neural Networks in C. `https://pjreddie.com/darknet/` (accessed on 2018-09-21).

Redmon, J. (2018, September). YOLO: Real-Time Object Detection. `https://pjreddie.com/darknet/yolov2/` (accessed on 2018-09-21).

Redmon, J., S. Divvala, R. Girshick, and A. Farhadi (2016, June). You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, pp. 779–788. IEEE.

Regier, T. and L. A. Carlson (2001, June). Grounding Spatial Language in Perception: An Empirical and Computational Investigation. *Journal of Experimental Psychology: General 130*(2), 273–298.

Roy, D. (2005, September). Semiotic schemas: a framework for grounding language in action and perception. *Artificial Intelligence 167*(1-2), 170–205.

Trieu, T. H. (2018, March). Darkflow. `https://github.com/thtrieu/darkflow` (accessed on 2018-09-21).

Xu, K., J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio (2015, February 11). Show, attend and tell: Neural image caption generation with visual attention. *arXiv arXiv:1502.03044 [cs.LG]*, 1–22.