Handling Anaphoras in multiple languages in the framework of Geometry Constructions

Jeetesh Mangwani & Pankaj Prateek
Advisor: Dr. Amitabh Mukherjee
Dept. of Computer Science and Engineering
Indian Institute of Technology, Kanpur, India
{jeeteshm, pratikkr, amit} @ cse.iitk.ac.in

April 17, 2014

_ABSTRACT

In this project, we take up the problem of handling anaphora in multiple languages in the context of a language-independent interpreter for drawing geometric diagrams. We focus on ruler and compass based construction problems. We start with use cases and motivations on why such a system would be useful and what places deploying it would be fruitful. We give a brief tabulated summary review of related research work done on geometry related problems and point out the common trait that they lack the ability to decipher any problem/constraint expressed in a natural language. We, then, move to describe the design of the interpreter and the usage of cross-lingual technique to provide language-independent interpretation ability. We briefly mention how is the alignment model utilised to realize the powerful translation feature. This is followed by the results obtained, difficulties faced and future work.

1 Objective

To design and implement interpreter that

- is language-independent (works for English, Hindi at present)
- Receives steps for a geometric construction as input e.g. "Draw a line segment AB of length 4 cm", "केंद्र B और त्रिज्या 5 सेमी लेकर एक चाप खींचिए जो पहले खींची चाप को C पर काटता हो" etc
- is capable of handling anaphoras
- Outputs the geometric figure obtained on executing the given sequence of steps

2 Introduction

It is often the case that students, teachers, architects and artists need to draw complex diagrams manually using simple geometric instruments like ruler, compass, set squares, dividers etc. This demands labor, time as well as expertise. Resorting to sophisticated graphics applications requires knowhow of application-specific details as well as expertise in coordinating hand micro-movements.

In order to save these resources required in drawing geometric diagrams as well as to reduce dependence on complex graphics applications, this project introduces an interpreter for diagram construction steps expressed in a suitable natural language. This not only simplifies the construction, but also leads to easily understood natural language 'programs'.

3 Related work

One of the most common techniques today to perform effective parsing is to use Part-of-Speech tagged language banks. Only few languages (about 8-10 of the 600+ languages in wide use) have the privilege of being resource-rich in the sense of having Penn-tree banks and other large-sized standardised corpus. An important observation is that fixed grammars, POS tags etc used in traditional parsers are not always the best way to break up a language. Taking a note of these shortcomings, we take a statistical approach to learn a language.

There has been a significant amount of work done on solving geometry construction problems. Gulwani et. al. [3] propose a method that uses goal-based heuristic to simulate backward deduction to solve a problem expressed in a predefined logical construct. Schreck et. al. [6] talk about the same problem but use CAD methods to deal with constraints. At the same time, Itzhaky et. al. [4] use the number of nondeterministic choices as a measure of a good solution. Ahmed, Umair et. al. [1] look more into using domain-specific measures to minimize parser errors and augment the geometry problem solver, GeoSynth. All these papers have provided us with valuable insights into selecting important and expressive constructs for our intermediate language.

We summarize our observations in terms of following 3 parameters:

- Uses other linguistic/domain knowledge
- Assumes linguistic cues are already translated into logical forms
- Uses Parse knowledge

Paper	Uses other linguistic/domain knowledge	Assumes linguistic clues already traslated into logical forms	Uses parse knowledge
Geometric Construction Problem Solving in Computer-Aided Learning [6]	YES	YES	NA
Synthesizing geometry constructions [3]	YES	YES	NA
Solving geometry problems using a combination of symbolic and numerical reasoning [4]	YES	YES	NA
Can modern statistical parsers lead to better natural language understanding for education? [1]	YES	NO	YES
Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars [7]	YES	NO	YES

Table 1: Related Works

Unlike all previous attempts, we do not assume that the logical forms of the sentences are available, nor we assume the availability of a parser. We use cross-lingual alignment to map the given sentence in natural language to metalanguage. It is therefore applicable to a large range of languages. We demonstrate this for two widely known languages belonging to very different families - Hindi and English.

4 Design

The interpreter consists of the following components:

- Aligner (training)
- Grammar

- Natural Language (NL) Interpreter
  - NL to Metalanguage Mapper
  - Heuristics-based Parser
  - Semantics Analyser
  - Plotter
- Using Context to handle Anaphoras

4.1 Aligner

Cross-lingual alignment is the heart of the learning mechanism in this project. It is used to obtain mapping/alignment between a natural language and our carefully designed and structured imperative metalanguage, L0. Since this alignment can be obtained for any natural language, the application is scalable to any number of natural languages. This technique is detailed in later sections

We have used GIZA++ [5] as the cross-lingual aligner. GIZA++ is a statical machine translation toolkit that is used to train IBM Models 1-5 and an HMM word alignment model.

4.2 Grammar

The grammar has been carefully designed to capture the underlying structure of the conventional (imperative) geometry construction steps. Note that there is a mild constraint on the grammar, which is a direct consequence of the assumption that the parameters of the object to be plotted occur near to parameter name in the input sentence. Hence the rules have been designed such that for each production rule, the leftmost non-terminal guides the search for the remaining non-terminals. [see section on Proximity]

4.3 Natural Language (NL) Interpreter

4.3.1 NL to Metalanguage Mapper

The interpreter component exploits the alignment model obtained from the Aligner component. Given a sentence expressed in a natural language, we use the alignment model to get the statistically most probable mapping of the NL sentence to a list of words in the language defined by the grammar. This list of words need not conform

to a sentence in the language defined by the grammar but still carries some intent of the NL sentence, and hence can be called a partially ordered list of metalanguage words.

4.3.2 Heuristics-based Parser

At this stage, the parser takes the partially ordered set of words generated in the previous step and tries to map it to a sentence formed by the grammar. This is done by performing a DFS on the grammar tree and trying to fit every token in the set with some grammar terminal. Given any production rule, our DFS-based algorithm recursively expands the leftmost unexpanded nonterminal in the rule; on encountering a terminal, it tries to fit the terminal to some token near its parent's/left sibling's token. The subtrees which are not satisfied are pruned and search resumes with the next available production rule. The search is aided by the following heuristics:

1. **Proximity**
   Proximity plays an important role in our parsing method e.g. we have assumed that the object that needs to be constructed would be near the "construct" word, and the length of a line segment would be near its name. Consider, as an example, "Construct a line segment AB of length 5 cm". Here, using our assumption we start searching for the object to be constructed near the "construct" word; it comes out to be the line segment AB. Note that we have to search on both the sides of the keyword (Give a hindi sentence as an example). This assumption is justified by the structure of the sentences in the natural languages, and specifically the geometry construction steps. This reduces the parsing complexity many-fold. As an another example, "Construct a line segment AB such that length of AB is equal to the difference of the length of CD and length of EF". Here, only those parse trees which demand the construction of AB are retained (as AB is nearer to "construct" than CD and EF); remaining trees which involve the construction of CD or EF are pruned.

2. **Type system**
   The system is implicitly type-defined. For example, consider a statement like "Mark a point A on it". Here "it" can be a line, line segment, circle etc. but not a point, as marking a point on a point is not permitted. Also, point pairs like AB would define a line segment or a ray, ABC would define an arc, 'l' would define a line (according to the conventions in the existing geometry texts). The grammar and the semantic analyser together form the type system.

3. **Validity of the output**
   The search is also guided by the output produced at this stage. If the output produced seems correct, i.e., it uses most of the tokens in the set and conforms to some sentence generated by the grammar, it is passed to the semantic analyser.

The semantic analyser then checks whether the output represents a valid intent to draw, i.e., a valid plottable object can be inferred out of it. If not, the parser tries to correct the parse tree starting from the rightmost leaf; this procedure is repeated until either we obtain something plottable or exhaust all the possibilities.

4.3.3 Semantics Analyser

This stage of the parser uses Lex-Yacc to generate a set of primitive objects like points, lines, arcs etc. which need to be plotted in the last input construction step. All coordinates, radii, lengths and angles are resolved in this step. Anaphora are also resolved in this stage. This is detailed in the later sections.

4.3.4 Plotter

For each input construction step, the plotter receives a list of primitive objects from the semantic analyser and plots them on the canvas.

4.4 Using Context to handle Anaphora

The semantics analyser uses the context (a list of objects which were plotted in the previous construction steps) to resolve the anaphoras. For example, "Mark a point M on it". Here "it" would refer to the most recently plotted markable object (objects on which a point can be marked e.g. lines, line segments, arcs, circles etc.). We fetch such markable object from the context to resolve the anaphora.

5 Cross-Lingual alignment

Cross-Lingual alignment is a technique to statistically align the words of a given pair of languages. It is close to translating one language into another, without any syntactic or semantic knowledge of any of the languages. As an example, for a pair of langauges, given sufficiently many sentences as illustrated in the table 2, a cross-lingual alignment assigns probabilities to the event that a particular source-language token is mapped to a target-language token.

English	Hindi	Meta Language
Construct a line AB of length 4 cm	4 सेमी लम्बाई का एक रेखाखण्ड AB खींचिए	`construct lineSegment AB length 4 cm`
With A as center and radius 3 cm, draw an arc	केंद्र A और त्रिज्या 3 सेमी लेकर एक चाप खींचिए	`constrcut arc center A radius 3 cm`
With B as center and radius 5 cm, draw an arc cutting the previously drawn arc at C	केंद्र B और त्रिज्या 5 सेमी लेकर एक चाप खींचिए जो पहले खींची चाप को C काटता हो	`construct intersectingArc center C radius 5 cm cuts arc previous at C`

Table 2: Sample Corpus

English	Meta Language	Probability
Construct	`construct`	1.00
Line segment	`lineSegment`	1.00
intersects	`intersect`	0.93
intersect eachother	`intersect`	0.78
cut eachother	`intersect`	0.87
cut	`cut`	0.98
join	`join`	1.00
mark	`mark`	0.98
label	`mark`	0.98
bisector	`bisector`	0.60
interior	`interior`	0.20
exterior	`exterior`	0.20

Table 3: Sample alignment between English and Metalanguage

Hindi	Meta Language	Probability
लगा दीजिये	`construct`	0.90
खींचिए	`construct`	0.98
रचना कीजिये	`construct`	0.96
बनाइये	`construct`	0.98
रेखाखण्ड	`lineSegment`	1.00
परस्पर प्रतिच्छेद	`intersect`	0.98
काटता हो	`cut`	0.82
काटे	`cut`	0.95
मिलाइये	`join`	0.98
जोड़िये	`join`	0.98
अंकित कीजिये	`mark`	0.98
मान लीजिये	`mark`	0.98
समद्विभाजक	`bisector`	0.59
अभ्यन्तर	`interior`	0.20
बहिर्भाग	`exterior`	0.20

Table 4: Sample alignment between Hindi and Metalanguage

6 What has been done

6.1 Corpus

The corpus contains around 350 sentences from the geometry construction field in both the languages. These were collected from the NCERT textbooks of standards 6^{th} to 9^{th}. English-Metalanguage and Hindi-Metalanguage sentence pairs were generated manually which were then used to train the cross lingual alignment software

The corpus has been formatted as follows:
For every sentence in natural language,

- First line corresponds to the natural language sentence.
- The second line corresponds to the intended meta langauge sentence.
- The third line is an empty line.

The corpus is available at [2]

To get an almost perfect probabilistic map between the parallel corpus, we impose a mild constraint: let the map be from language A to B; we design the corpus such that every word of the language A either maps to exactly one word of language B or to the

empty word. Note that this is not necessary. It has been done so as to obtain the best possible map for a given corpus size under corss-lingual alignment. Poorer quality of map leads to considerable increase in the possible translations that the parser would have to parse.

6.2 Parser

Suppose the input sentence is "Construct a line segment AB of length 7 cm". Using the English-to-Metalanguage map, we translate this sentence tokenwise to "construct line segment AB length 7 cm" where “a” and “of” map to empty words. Similarly, for the input sentence “7 सेमी लम्बाई का एक रेखाखण्ड AB खींचिए”, using the Hindi-to-Metalanguage map, we obtain the translation “7 cm length line segment construct”, where “का” and “एक” map to empty word.

For both of these translations, we obtain the same parse tree, which is illustrated below:

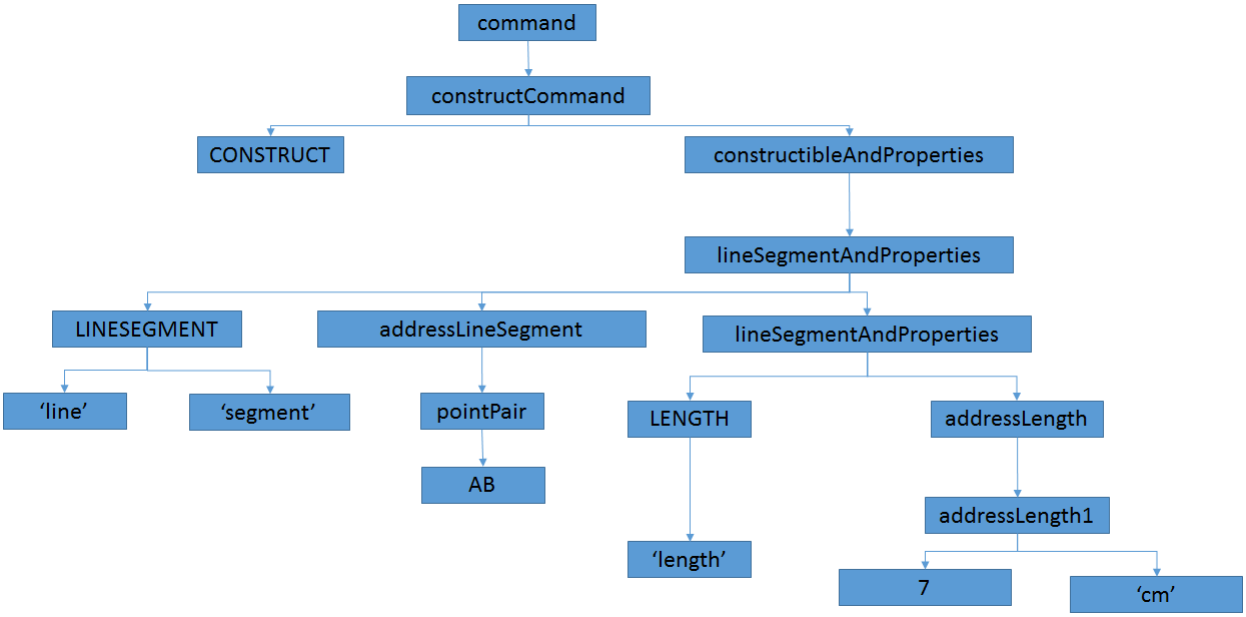

Figure 1: Sample Parse Tree

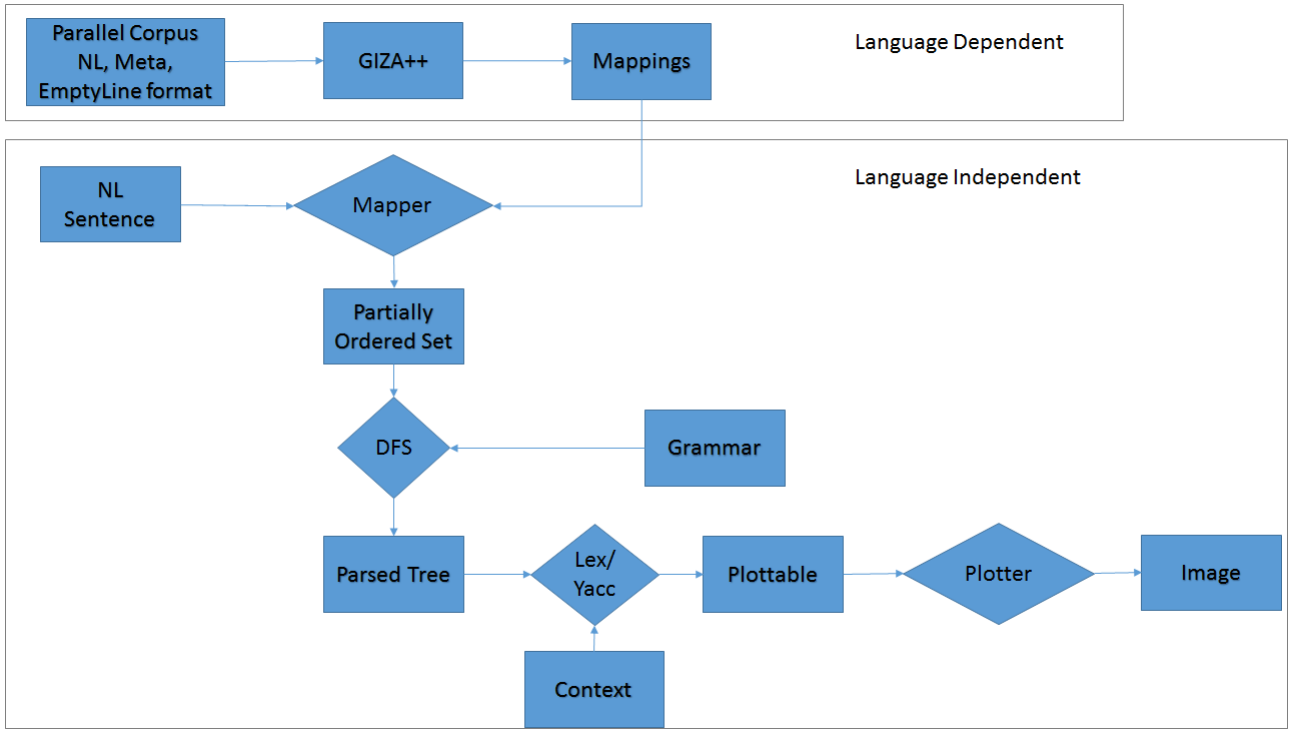

Figure 2: Work Flow

This approach has been shown to work for simple construction steps. Output diagram for the steps above is shown in Figure 2.

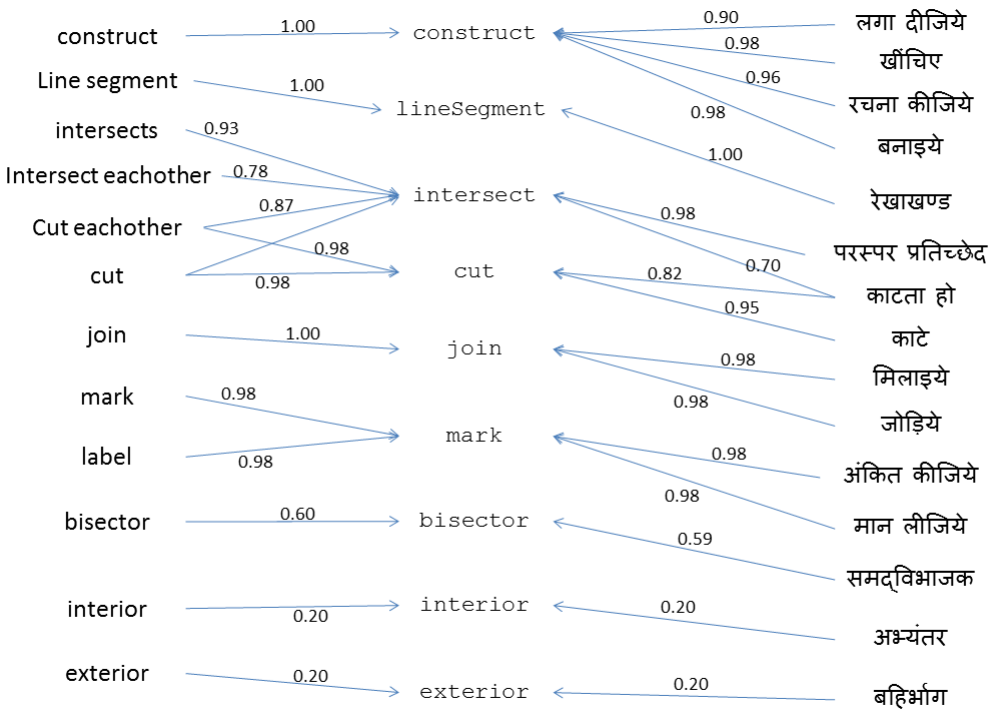

Figure 3: Graphical visualization of the sample alignment

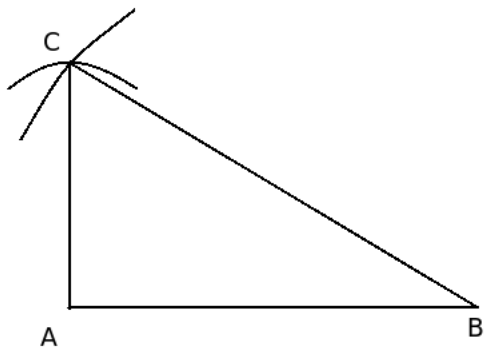

Figure 4: Constructed Triangle

7 Results

1. Number of sentences in the corpus: 360 each in English-to-Metalanguage and Hindi-to-Metalanguage corpus
2. A sample prototype implementation that demonstrates the performance of the system for simple construction steps of drawing line segments and arcs.
3. Number of unique tokens in each of the three languages:

Language	**Number of unique tokens**
English	181+
Hindi	169+
Metalanguage	110+

Table 5: Unique tokens in different languages

7.1 Difficulties

- **Anaphoras** Dealing with anaphoras like "this", "it", "previous", "last" etc in sentences like "Mark any point M on it", "एक सुविधाजनक त्रिज्या लेकर पिछले चरण वाले चाप को बिन्दु A पर काटें।' etc
- **Underspecified Parameters** We also need to provide for unspecified radii, lengths and anlge measures e.g. "With A and B as centers and a suitable radius, draw two arcs interseting eachother at point C"
- **Probabilistic Mapping** Since the mapping is statistical in nature, there might be various close alignments for the same source-language word. This forces us to resort to try out different possible alignments and look for the one that fits e.g. for the sentence "Construct AB of length 7.8cm", we get the following alignments with their corresponding probabilites (we assume that probabilites of word-wise alignment are independent of eachother)

Mapped meta-language sentence	**Probability**
construct AB any length 7.8cm	0.716853
construct AB lineSegment length 7.8cm	0.21081
construct AB angle length 7.8cm	0.0723206
construct AB center length 7.8cm	1.90645e-06

Table 6: Map to possible metalanguage sentences alongwith their probabilities

8 What needs to be done

1. Enrich the corpus to cover larger domains of expressions e.g. having larger number of primitive steps, providing for higher-level primitives, specifying relative distances, dealing with anaphoras and references

2. Expand the functionality of the system to work for all kinds of construction steps claimed so far

3. Deploy the system on a webserver to demonstrate its functionality and work towards making it more robust

4. In order to capture colloquial (non-academic) ways of expressing construction steps, we need to develop a user interface that augments the corpus with input sentences

References

[1] Umair Z Ahmed, Arpit Kumar, Monojit Choudhury, and Kalika Bali. Can modern statistical parsers lead to better natural language understanding for education? In *Computational Linguistics and Intelligent Text Processing*, pages 415–427. Springer, 2012.

> Focusing on the educational domain, this paper seeks to build robust domain specific natural language understanding modules by suitable preprocessing of the text and use of domain knowledge to deal with parser errors. The paper starts with showing that even off-the-shelf parsers do not perform up to the mark on texts with domain-specific vocabulary and long sentences. The paper also talks about GeoSynth, built using state-of-the-art parsing technology. GeoSynth is a system which can automatically solve geometric constructions word problems.

[2] Parallel Corpus for Geometry Constructions. http://cse.iitk.ac.in/users/pratikkr/cs498/index.html.

[3] Sumit Gulwani, Vijay Anand Korthikanti, and Ashish Tiwari. Synthesizing geometry constructions. In *ACM SIGPLAN Notices*, volume 46, pages 50–61. ACM, 2011.

> Aiming to solve ruler and compass based geometry construction problems, the tool described in this paper claims to have synthesized construction steps for problems posed from high-school textbooks and examination papers. It focuses on three important insights viz. reduction of symbolic reasoning to concrete reasoning using verification based on random testing,

augmenting the instruction set with higher level constructs and simulating backward deduction used by humans by using a goal-based heuristic.

[4] Shachar Itzhaky, Sumit Gulwani, Neil Immerman, and Mooly Sagiv. Solving geometry problems using a combination of symbolic and numerical reasoning. Technical report, Technical report, Tel Aviv University, 2012.

This paper combines deductive, numeric and inductive reasoning to solve geometric problems. Using a mixture of these methods, several problems are solved which, otherwise, wouldn't have been solved using the methods individually. The paper uses the number of nondeterministic choices as a measure of how close a partial program is to the solution, claiming this to be useful in grading and providing hints to the students. The implementation takes an average of few seconds on 18 Scholastic Aptitude Test geometry problems and 11 other ruler and compass-based geometry construction problems.

[5] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51, 2003.

[6] Pascal Schreck, Pascal Mathis, and Julien Narboux. Geometric construction problem solving in computer-aided learning. In *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*, volume 1, pages 1139–1144. IEEE, 2012.

This paper introduces CAD methods to deal with Constraint Satisfaction Problems related to geometry, at the same time claims to have met the requirement met by the educational domain. The methods show that results must be construction programs to take into account particular cases. A knowledge-based system implemented in Prolog has also been presented.

[7] Luke S Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*, 2012.

With the objective of mapping natural language sentences to their counterpart lambda-calculus encoding, this paper introduces an algorithm which learns from the input set of natural language sentences labeled with lambda calculus expressions. The algorithm attempts to infer the underlying distribution over semantic and syntactic analyses conditioned on the input sentences and outputs a log-linear model.