Welcome to the fifth workshop on controlled natural languages in Aberdeen. This is the fourth time that I've visited the conference, the third time that I've spoken at the conference and the first time that I am invited to speak. It's an honour to be the first speaker of the day and to be asked by the organisation to deliver a key note presentation. That makes me feel like a special guest. But actually I always felt a bit like a special visitor since I do not work for a research institute while most of you do.

That is not to say that I don't DO research. Actually that's why I am here. To learn new things and to be inspired. Today I would like to share with you what I have learned from you, you as in the CNL community, what I was looking for, and what I did not find.

I will also take the time to demonstrate what I have done with the knowledge I gathered. I have two companies, one for services and one that offers a software product. Actually what I have learned here is an important part of how I earn my money.

About my background

But let me begin with my background. Like many others at the age of eighteen, I did not know what to study when I finished college. I liked many things so I decided to choose an interdisciplinary study consisting of many things.

The study combined philosophy, psychology, mathematics, information science, biology and linguistic. It was all very new and named artificial intelligence. Nobody really understood what it was about. My parents did not believe I could make a living after I graduated and I did not expect an easy transition to a working life. The youth unemployment rate in those days was 26%. That was my reality.

The focus in University was on logic. My professors were old; one of them is Dirk van Dalen, now

having a Wikipedia page. He worked in the tradition of Brouwer and we spent a lot of time thinking about modalities and possible world semantics. I liked the way you could get submersed in these topics. And of course I liked the fact that the results were indisputable by nature. A black and white world.

Computational linguistics and Lambda calculus were two of the most boring and difficult subjects. The subjects were popular though, probably because of the professor who was a very handsome man with excellent presentation skills, named Michael Moortgat.

And of course we learned programming in the following order: functional programming (lisp), declarative programming (prolog) and procedural programming (pascal). Later they changed this order and had students start with a procedural language. The results were that the grades for functional and declarative programming dropped dramatically.

We learned these language by starting with the grammar. It was like learning a language. Very different from todays "how to … " books for dummies. Hence we learned how to use context free grammars and for building our expert systems we made our own context free grammars.

About my work environment

The typical organisation I work for is an administrative organisation whose products and services depend on a non-standard and complex decision making process. The decisions have been made by humans in the past (named subject matter experts) or systems that were created a long time ago (named legacy systems). The decisions typically involve heuristics that are gained by experience or must be compliant with some policy.

An example is an organisation that distributes permits. Other examples are claims processing by insurance companies, mortgage eligibility by financial institutions, social benefits, tax administration, clearance for a ship to enter a harbour etc.

These organisations face a couple of issues. Most of them are not able to guarantee consistency in the operation across different departments because each department interprets policy independent of each other. When processes are automated we see that IT has been in the lead for years resulting in specifications that are difficult to validate, change and maintain by the business.

Non-administrative organisations have the same issues by the way. Rules for traffic management, crowd control and fraud detection must be traced to policy, understandable and explain themselves.

All this results in issues with automation. Systems are difficult to change and are blocking the organisation's capability to innovate. The magic word is AGILITY. My assignments are all related to helping organisations to become more agile.

I define agility as an emerging behaviour that is related to traceability, manageability and automate-ability. When all are good your agility is optimal. But very often you need to find a balance between these aspects. For example: I need natural language because it's more manageable. I need decision tables and declarative logic because 'they're more automate-able. I need atomic rules because 'they're easier to trace to policy although 'they're not good for manageability. While maturing and becoming an expert myself I see all the nuances of getting

to a good solution.

On logical consistency

I don't believe anymore in the 'ultimate' solution. We started naive in the field of expert systems. First of all they thought the expert himself could make his own knowledge explicit which is like a snake eating its tail. We had this dream that there would be a constant feedback loop between the computer and the expert. The system is to be feed with knowledge by the expert, the system can explain it's reasoning, the expert can then correct the knowledge when needed. They would learn from each other to perform better as a team. But in reality I don't know any expert who was willing and able to realise this dream. I tried to help them. But after passing the first hurdle of understanding the difference between a boolean and a variable and a rule the next hurdle was logical. The created rules were redundant, inconsistent and incomplete.

So I started with LibRT to make a verification engine named VALENS that could check a set of rules on logical redundancy, incompleteness and consistency. It turned out that IT was not interested (they never make such mistakes …) but the business was. They wanted to know if their rules were consistent and complete. We were happy with their request but our technology could not help them…. their rules were prose in a word document, legalise in a PDF document or something that we would now call 'user stories'.

On structured language

Around that time I learned about the business rules community. They were interested in my work on verification, validation and more structured methods like decision tables. I was interested in their ideas of structured natural language to write rules and involve the business. We worked on RuleSpeak and SBVR as a way to standardise the meaning of business rules.

But I quickly realised that the big cap in the market was a tool for the business to support them in writing and managing those rules. Customers were (and some are still) trying to be consistent and complete in Microsoft Word or Excel. Although I know some human parsers (and some of them are probably in this audience) … I am not a human parser and the employees of my customers sure aren't. So we needed a tool for the business and that's when we started to develop RuleXpress and I made my first visit to the CNL workshop.

I was looking for a way to express guidelines and patterns for writing rules, independent of a specific natural language. Guidelines like a rule must start with a subject, the subject should not be in plural and one should not combine disjunctions and conjunctions in one rule.

I did not want to program these guidelines for each language again. Could I express these guidelines in a meta language? Did a component exist that would transform multiple natural languages to this one meta language? The first workshop the answer was no. The second workshop I visited I met Aarna Ranta and the answer was 'yes' or at least 'in theory yes'.

Meanwhile we were working on the development of RuleXpress, got more customers and gained a better understanding of the challenges. Let me tell you a little more about the tool and the company.

More about RuleXpress

RuleXpress is developed by RuleArts. RuleArts is owned by Business Rules Solutions and LibRT. We have 50 customers licensing 500seats. Our customers are very diverse and working in multiple domains. Our users are in the business. This makes it difficult to acquire and install software. The IT department that is responsible for software purchase does not understand why Microsoft word and Excel are not good enough or otherwise why you should not use some of the tools they use for database design and system architecture.

My users are new to working in a more structured environment and when I constrain them too much they run back to the office tools. So the tool has no syntax, no constraints, is methodology independent and fully configurable. Many users do not follow a training and find their own way in the product (although we prefer our customers to follow training).

Although all customers use natural language to write rules they also use many words that are not common or not in a dictionary. The consequence is that all language support technologies that are trained on a general corpus are not useful for me. Most of the words my customers use are not in the corpus. So RuleArts had to develop its own language support technology features and there is little help of standard tools and libraries.

What I learned from the CNL community

One of the first language related features that we realised is the recognition of defined nouns or noun phrases in sentences when written in singular and plural form. Users manage a vocabulary defining nouns or noun phrases in singular form. For each language (English, Spanish and Dutch) we created a set of heuristics that generates the plural form based on the singular term. Coverage differs per language, Dutch being the worst one with 85%. But we and our customers are happy with this strategy because:

- it works for all words, also words that do not exist in the dictionary.
- it is very effective for our data storage and data footprint
- exceptions can be handled by the end-user

Furthermore our algorithm to recognise words will, in case of ambiguity, find the biggest match from left to right.

The second feature is to make sure similar kinds of rules are written in a similar way. I learned here at CNL that a context free grammar is probably good enough for the kinds of sentences we need to parse. Characteristics of context free grammars are well known and they can be processed quickly. For the specific case that we need anaphoric references we use a special grammar element with special meaning.

The third challenge was to support the context free grammar in the user interface. I did a small research project with the University of Nijmegen. The objective was to find out who performed better: rule authors that write rules in a free form text editor and some guidelines or rule authors working in a predictive editor based on a grammar. The result were in favour of the free form text editor. It turned out that rule authors using the predictive editor did not correct themselves if they started in the wrong way.

So I wanted to make a match algorithm that indicates if a rule matches a grammar pattern or if there is a partial match with a pattern.

> *Given this context I use the word CNL, context free grammar and pattern as synonyms from now on.*

Inspired by examples from Rolf Schwitter who showed the strength of regular expressions in this workshop we decided to experiment with generating regular expressions from the context free grammar. We were afraid that the performance would not be good enough but in combination with some tricks we receive a very acceptable performance.

The latest feature is that we recognise which defined associative relationships are used in a sentence. The relationships between concepts are defined in advance. They are bidirectional and can be expressed as a fact type sentence or drawn in a diagram.

The algorithm to find matching fact types in a rule sentence makes pairs of matched terms in the sentence and looks at the words between those concepts. The words between those concepts should make a match with the relationship symbol of the fact type, eventually in a different tense. We use a generally available stemmer for multiple languages (among others English, Dutch and Spanish).

There are different kinds of relationships between concepts. We use the list of SBVR and use some of its defined semantics to make the match smarter. For example we can find matches based on specialization relationships where relationships are inherited from more general concepts.

More happy customers

Typically our customers grow while using the tool. In the first stage they are very happy that the Excel sheet is replaced with something more manageable and that the glossary automatically matches in the rule statements. After 2 years of working to improve the organisation of the rules and the vocabulary the next step can be made. They work with a team and need to make sure that similar knowledge is written in the same way. So they standardise all kinds of keywords. But when patterns are getting more complex and include other verbs than modal verbs they need the conceptual model.

I am working in a field that changes its name every five years. While I still do the same I had to present myself as an Expert systems programmer, RuleBased system, Knowledge engineer, Knowledge manager and business rules analyst. Today I should be doing decisioning or decision management but I thought I am old enough now to stop following trends. I did notice that it is allowed again and even ‘hot’ to do artificial intelligence so I added that label recently to my LinkedIn profile.

> *What's in a name?*

Recently a prospect asked when RuleXpress would be named 'DecisionXpress' and I answered 'never'. The answer was considered 'brave' in a world that is dictated by hype cycles.

Actually my answer was a little longer. I answered that RuleXpress would never be named DecisionXpress because it would narrow our market to an even smaller market and we already support decisions. Instead I would like to broaden our market and name it 'LanguageXpress'. The idea of structured language based on patterns is also useful for requirements, user stories, questions, simple agreements …. and I believe many more that we will find when we make it.

The biggest challenge to realise this vision is that my users are lawyers, traffic engineers, accountants, policy makers, tax administrators, physicians, green energy specialists and the like… and they can not create a context free grammar.

But the situation is even worse…. I thought that most programmers would know how to create a context free grammar…. but the don't. In the time that languages and interpreters had to be created these were important courses in the curriculum of university and high school. But today we only educate users of those languages.

Future for controlled natural languages

There is a huge opportunity for controlled natural languages in organizations. It's a way to involve the business and generate code, brochures, explanations, work instructions and many more all from the same source. These will all be based on a single (or the same) interpretation of the policy and thus organizations become more consistent and agile in their operations. We need educated people who are able to create the controlled natural languages (patterns) and perform transformations to different domain specific languages or presentation formats. The research community we call that (natural) language generation.

An alternative for hand-written CNL's is that we generate them based on a set of example sentences. A user may guide the process and indicate which elements in a sentence must be fixed, variable or optional. When you already have a set of sentences (and most of my customers have) you can show all other sentences that meet the pattern and ask which of the pattern matches are incorrect. Will the system and the user work well together and find the right generic pattern? Could we make a real expert system that works together with my users to make a context free grammar without bothering my user with stupid questions? I would like to leave this for a next research project.

The biggest surprise for me is that I did not need the methods of computational linguistics to come this far but I did need the knowledge offered by the field of computational linguistics to understand.

In this community I have always asked questions on how to evaluate a controlled natural language. I thought this to be important to get acceptance for controlled natural languages. But after writing this speech I believe it's less important for me than I thought. When I create patterns for my users that they don't understand or don't like then they will not use it. It's as

simple as that.

I am open to any other questions now. Thank you for your attention.

Let me know if you learned something new by sharing this post.

