Tutorial on Modeling VAT Rules Using OWL-DL

Nielsen, Morten Ib; Simonsen, Jakob Grue; Larsen, Ken Friis

Publication date:
2007

Document version
Publisher's PDF, also known as Version of record

Tutorial on Modeling VAT rules using OWL-DL

Morten Ib Nielsen, Jakob Grue Simonsen and Ken Friis Larsen
Department of Computer Science
University of Copenhagen
Email: {mortenib|simonsen|kflarsen}@diku.dk

August 28, 2007

Total number of pages: 16

Abstract

This paper reports on *work in progress*. We present a methodology for constructing an OWL-DL model of a subset of Danish VAT rules. It is our intention that domain experts without training in formal modeling or computer science should be able to create and maintain the model using our methodology. In an ERP setting such a model could reduce the Total Cost of Ownership (TCO) and increase the quality of the system. We have selected OWL-DL because we believe that description logic is suited for modeling VAT rules due to the decidability of important inference problems that are key to the way we plan to use the model and because OWL-DL is relatively intuitive to use.

1 Introduction

Imagine an ERP system where domain experts can create and implement changes in e.g. VAT rules without the help of programmers. The benefits would be shorter development time and fewer mistakes due to misinterpretation of specifications which lead to reduced TCO and increased quality of the software. On a coarse-grained scale such a system consists of three parts: A model of the rules, a tool to edit the model and the core ERP system using the model. In this paper we focus on the first part - the model. A priori two requirements exist. First the modeling language must be strong enough to express the rules in question and second it must be easy to use without training in formal modeling or computer science. In a more general setting the model can be used as a VAT knowledge system which external programs can query through an interface. In the long run we envision that authorities such as SKAT (Danish tax administration) can provide online access to the model e.g. using web services such that applications always use the newest version of the model.

In this paper we describe a methodology we have used to develop a model of a subset of Danish VAT rules using the general purpose Web Ontology Language (OWL) editor Protégé-OWL[1] and we report on our experiences in doing so. We selected a subset of Danish VAT rules consisting of flat VAT (25%) plus a set of exceptions where goods and services are free of VAT, chosen because they seem representative. Further the rules are accessible to us by way of an official guideline by the Danish tax administration. Our study is focusing on the feasibility

[1] `http://protege.stanford.edu/overview/protege-owl.html`.

of using OWL to model VAT rules and not on the usability of the Protégé-OWL tool itself. By feasibility we mean how easy or difficult it is (for a human) to express and understand VAT rules in OWL, in particular this does not cover issues such as modularization. The methodology presented here is inspired by the article [1] together with our own experience. Readers of this guide are assumed to have user experience of Protégé-OWL corresponding to [2] but not of computer science nor of modeling in general.

1.1 Motivation

One of the overall goals of the strategic research project 3gERP is to reduce the TCO of Enterprise Resource Planning (ERP) systems. We believe that a VAT model helps to this end in two ways. First we envision that domain experts create and update the model thus eliminating a layer of interpretation (the programmer) where errors can be introduced. Second a VAT model can change handling of VAT from being a customization task into being a configuration task, meaning that no code needs to be changed when the model is updated.

VAT and legal rules in general deal with frequent transactions between legal entities. Transactions are typically triggered when certain conditions are fulfilled and therefore dynamic checks on these conditions are needed. The idea is to use the model to automatically infer what actions should be taken based on the conditions. In the case of VAT rules we can ask the model whether a delivery is subject to VAT or not based on the information we know about the delivery. The answer from the model will be *Yes*, *No* or *Maybe*[2] and can be used to trigger an appropriate transaction. In a broader perspective the model is supposed to work as a VAT knowledge system that given a context and a question can tell other systems what to do, e.g. guide accounting systems and if required indicate that authorities should be contacted etc.

1.2 Roadmap

The remainder of this paper is structured as follows. In Section 2 we give a short account of description logic and OWL. In Section 3, 4 and 5 we present our methodology by giving examples. Finally we outline future work in Section 6 and we conclude in Section 7.

2 Description Logic and OWL

In this section we give a short introduction to description logic (DL) and OWL. This introduction can be skipped, if you are already familiar with the concepts. Description logics are knowledge representation languages that can be used to structure terminological knowledge in knowledge systems which are formally well-understood. A knowledge system typically consists of a knowledge base together with a reasoning service. The knowledge base is often split into a set of concept axioms the *TBox*, a set of assertions the *Abox* and a *Role hierarchy*. These constitute the *explicit* knowledge in the knowledge system. The reasoning service is a program that can check the consistency of the knowledge base and make implicit knowledge explicit, e.g. decide equivalence of concepts. Since the reasoning service is a pluggable component knowledge systems separate the technical task of reasoning from the problem of constructing the knowledge base.

[2] In the case where insufficient information is provided in order to answer the question.

2.1 OWL

OWL which is short for Web Ontology Language is an ontology language designed to be compatible with the World Wide Web and the Semantic Web. The most important abstraction in OWL is concept axioms which are called classes. Each class has a list of necessary conditions and zero or more equivalent lists of necessary and sufficient conditions [2]. A list of necessary conditions is a list of conditions that every member of the class must satisfy. In the same way a list of necessary and sufficient conditions is a list of conditions that must be satisfied by every member of the class and if satisfied guarantees membership in the class. OWL is based on XML, RDF and RDF-S and can be used to represent information in a way that is more accessible to applications than traditional web pages. In addition OWL has a formal semantics, which enables logic reasoning. OWL comes in three variants: OWL-Lite $\subseteq$ OWL-DL $\subseteq$ OWL-Full of increasing expressive power. The variants OWL-Lite and OWL-DL are based on the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ respectively [3], which guarantees that important inference problems such as satisfiability and subsumption are decidable. Since OWL is XML based we need an editor to create OWL ontologies. We have used the general purpose OWL editor Protégé developed by Stanford Medical Informatics at the Stanford University School of Medicine.

3 VAT Exemption 1: Sales outside EU

Our methodology is aimed at modeling VAT rules as described in guidelines instead of the raw law text itself. This choice was made because guidelines are more accessible to us, and because these are the rules that small companies adhere to in practice. Further the investigation of the feasibility of using OWL to model VAT rules concerns the ease with which rules can be formalized and not so much from where the rules are extracted[3]. In what follows we refer to the guideline as the *legal source*.

In order to ease reading we have used the word *concept* only when we speak about the legal source. The corresponding concept in the model (OWL) is called a *class*. A concept in the legal source is modeled as one or more classes in the model.

Here we present the steps we took in order to make our model of Danish VAT rules.

3.1 Pre-modeling

1. Download Protégé-OWL from `http://protege.stanford.edu/download/release/full/` and install. Make sure you can start Protégé in OWL-mode (logic view). When started and if you select the *Class* tab it should look like Figure 1.

2. Download [2] and read it. This is **important** because many of the constructions we use are explained herein.

3.2 Modeling

First you must decide which legal source(s) you want to model.

[3]Since we have used the official guidelines by SKAT (Danish tax administration) we believe that the content of the guidelines is in accordance with the law.

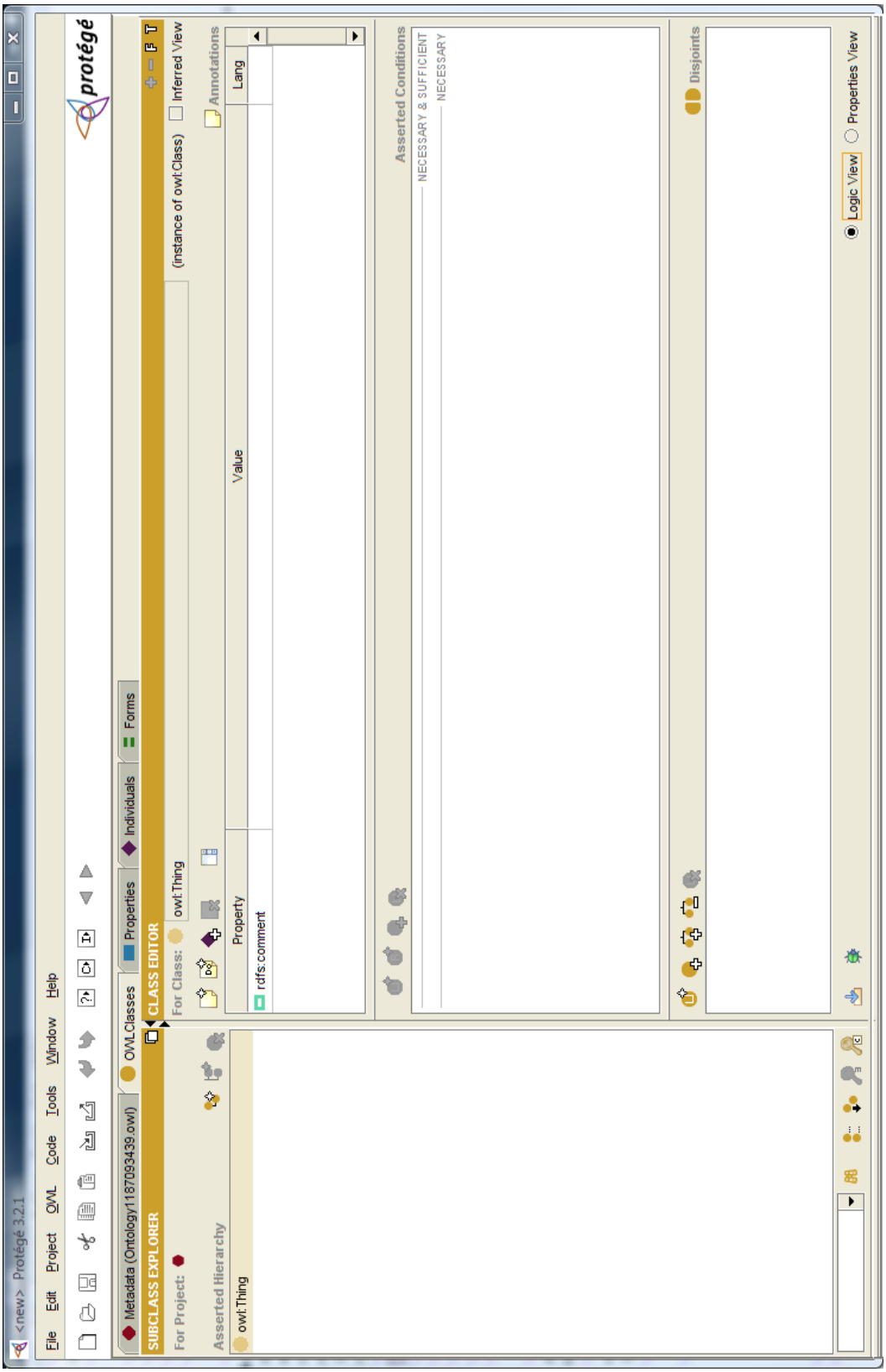

Figure 1: Protégé-OWL class-tab, logic view.

In our case we used the official guideline *Moms - fakturering, regnskab mv, E nr. 27, Version 5.2 digital, 19. januar 2005.*

3.2.1 Overall framework

Modeling should start with a read through of the legal source. Based on this general (to be refined later) classes such as `Location`, `Goods`, `Services` and `FreeOfVAT` together with attributes such as `hasDeliveryType` and `hasSalesPrice` can be created as subclasses of the built-in top-level class `owl:Thing`. An attribute can usually take on at most a finite number of values. In that case we use value partitions to model them as described in [2][p. 73-76][4]. If the domain is not finite we use data type properties instead. Deciding on the overall framework helps to structure the capturing of rules in a homogeneous way and enables working in parallel (which can be needed if the legal source is large). After our read through of the legal source we arrived at the overall framework in Figure 2.

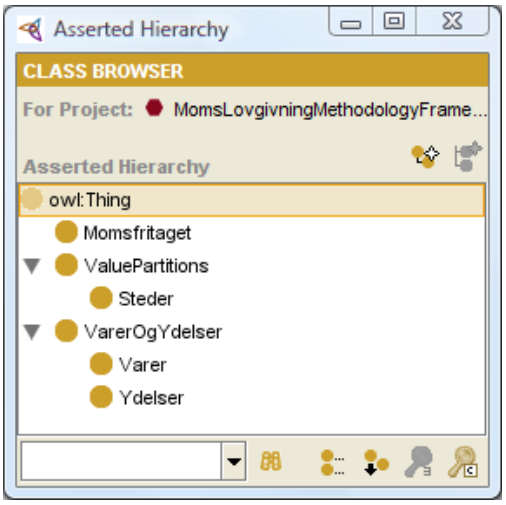

Figure 2: Overall framework.

`Naming Convention.` All classes, properties, individuals etc. should be given names picked from or inspired of the legal source. All names should be in the same language as the legal source (in our case Danish). Using the naming convention supported by Protégé-OWL class and individual names should be written in Pascal Notation, e.g. *InternationalOrganization* not *internationalOrganization* or *International_Organization*, while property names are written in Camel Hump Notation, e.g. *someProperty*. Typically a property is used to assign an attribute to a class. In this case we prefix the name of the property with a verb describing the kind of relation the class has along that property, e.g. *hasNumberOfSides* or *isFragile*.

3.2.2 Rule modeling - step I

Having modeled the overall framework it is time to go through the legal source one section at a time looking for rules that should be modeled. Here we give an elaborate description of how to model a single rule from the legal source starting from the overall framework in Figure

[4] An exception is the domain of truth values, which is built-in as a data type.

Table 1 Extract from the legal source and its translation into English.

Salg til lande uden for EU (3. lande). Du skal ikke beregne moms af varer, du leverer til steder udenfor EU eller til Færøerne og Grønland. Det samme gælder normalt også for ydelser, men du skal dog opkræve moms af visse ydelser.

[4][p. 9]

And translated into English:

Sales outside EU (3rd countries). No VAT should be added to goods delivered to destinations outside the European Union, or to the Faroe Islands or Greenland. This fact ordinarily also applies to services, but VAT should be added to certain services.

Translated from [4][p. 9]

Table 2 Necessary & sufficient conditions for application of the rule in Table 1.

- The rule concerns sales.
- The rule concerns both goods and services.
- The place of delivery must be outside the European Union, or the Faroe Islands or Greenland.

2. In Section 4 and 5 we give a brief description of how to model other rules. Together the modeling of these rules cover all the constructions we have used in our VAT model. Since our legal source is in Danish we present the rules in their original Danish phrasing together with a translation into English. Now let us consider the rule shown in Table 1.

Since our model is only a prototype we make a slight simplification and assume that the rule also applies to *all* services. With this simplification we can identify the necessary and sufficient conditions for application of the rule. These are shown in Table 2. In order to model the necessary and sufficient conditions in Table 2 we must add some attributes to `VarerOgYdelser`. The first and second condition in Table 2 tell us that we must be able to model that goods and services are sold[5]. We do that by adding an attribute to the class `VarerOgYdelser` (translates into `GoodsAndServices`) which already exists in our overall framework. Attributes are modeled using functional properties. In accordance with our naming convention we select the name `harLeveranceType` (translates into `hasDeliveryType`). Since there is a finite number of delivery types we model this attribute as a *value partition*, i.e. an enumeration. Value partitions can be created using a built-in wizard[6]. Just as in [2] we store value partitions as subclasses of the class `ValuePartitions`. The reason plain enumerations are not used is that they cannot be sub-partitioned. Using value partitions we retain the possibility of further refining the concepts the value partitions model.

[5]Instead of being sold goods can also be used as e.g. a trade sample. See [4][p. 8-9] for other examples.
[6]Menu►Tools►Patterns►Value Partition....

Remark. Technically enumerations are constructed by defining a class in terms of a finite set of individuals plus a functional property that has this class as its range. Since individuals are atoms they cannot be subdivided. On the other hand a value partition is defined using a functional property having as its range a class defined as the union of its subclasses all of which are distinct. These subclasses can (because they are classes) be partitioned into more subclasses if needed.

Having created the value partition `harLeveranceType` which can have `Salg` (translates into `Sale`) as a value we need to add it as an attribute to the class `VarerOgYdelser`. This is done by adding to the *necessary conditions* an existential quantification over the corresponding property having the value partition (or data type in case of data type attribute) as its range. Thus we add ∃ `harLeveranceType some LeveranceType` to `VarerOgYdelser`. The third condition tells us that we must be able to model that goods and services have a place of delivery. A read through of the legal source tells us that only three places are needed namely *Denmark*, *EU* and *non-EU*. Thus this attribute which we name `harLeveranceSted` (translates into `hasPlaceOfDelivery`) must be modeled as a value partition.

Having modeled these attributes the class `VarerOgYdelser` looks as shown in Figure 3.

3.2.3 Rule modeling - step II

Now we are ready to model the rule itself. Since the rule describes a situation where you do not have to pay VAT we model it as a subclass of `Momsfritaget` (translates into `FreeOfVAT`). Following our naming convention we name the class `MomsfritagetSalgAfVarerOgYdelserTilIkke-EU` (translates into `VATFreeSalesOfGoodsAndServicesInNon-EU`). Then we add a textual description of the rule and a reference to where in the legal source the rule stems from to the `rdfs:comment` field. Next we must specify *necessary and sufficient* conditions on membership in `MomsfritagetSalgAfVarerOgYdelserTilIkke-EU`. It is important to remember that if a class has two sets of necessary and sufficient conditions then they must imply each other, see [2][p. 98]. Based on the necessary and sufficient conditions captured in Table 2 we add the following necessary and sufficient conditions to `MomsfritagetSalgAfVarerOgYdelserTilIkke-EU`:

- `VarerOgYdelser`
- ∃ `harLeveranceSted some Ikke-EU`
- ∃ `harLeveranceType some Salg`

The result is shown in Figure 4.

4 VAT Exemption 2: Sales to Embassies

In this section and onwards we will not mention when to add references to the legal source in `rdfs:comment` fields of classes and properties. The rule of thumb is that this should always be done. Now let us consider the rule in Table 3. We identify the necessary and sufficient conditions for application of the rule. These are shown in Table 4.

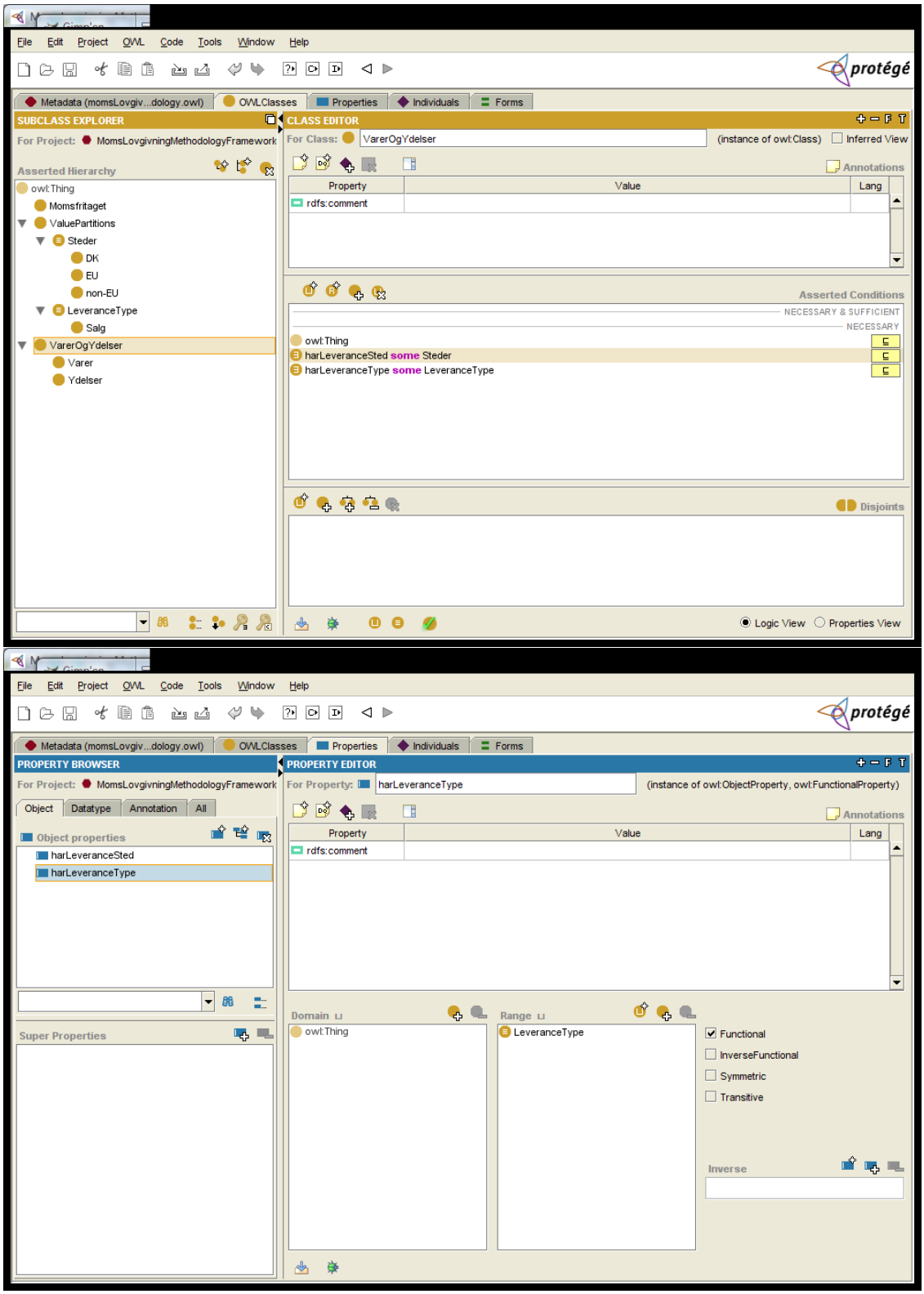

Figure 3: Class and property view after adding attributes.

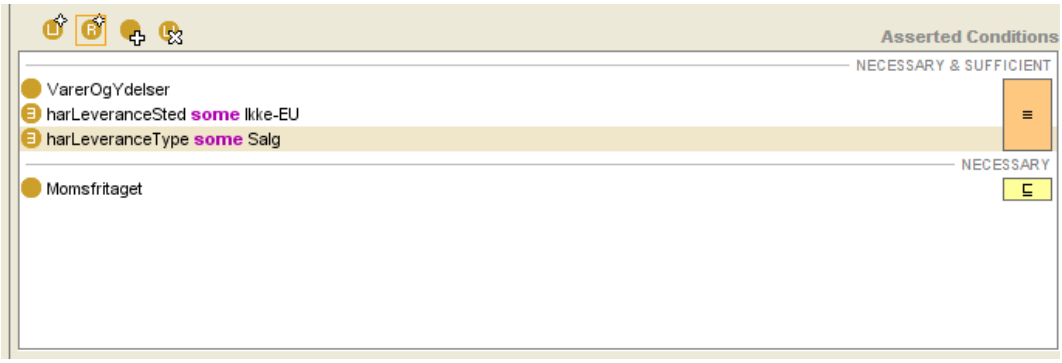

Figure 4: Asserted Conditions of our model of the legal rule in Table 1.

Table 3 Extract from the legal source and its translation into English.

> *Salg til ambassader. Du skal ikke beregne moms af varer og transportydelser, som du leverer til ambassader og internationale organisationer i andre EU-lande.*
>
> [4][p. 9]

And translated into English:

> *Sales to embassies. VAT should not be added to goods and transport services delivered to embassies and international organizations in countries within the European Union.*
>
> Translated from [4][p. 9]

Table 4 Necessary & Sufficient conditions for application of the rule in Table 3.

- The rule concerns sales.
- The rule concerns goods and transport services.
- The place of delivery must be in the European Union.
- The buyer must be an embassy or an international organization.

4.1 Rule modeling - step I

We are already able to model that the rule concerns sale and that the place of delivery must be in EU. We cannot model the specific service transportation yet. Therefore we must add it to our model. Since it is a service it should be modeled as a subclass of `Services`. We name the class modeling the service transportation `Transport` (translates into `Transportation`). Now we can model that something belongs to the set of goods and transport services by requiring membership of `Varer` $\sqcup$ `Transport`. Finally we must be able to model that the buyer is an embassy or an international organization. Since there are only finitely many different kinds of buyers we model this as a value partition, and because this attribute applies to both `Varer` and `Transport` we add it to their most specific common super-class which is `VarerOgYdelser`. We name this attribute `harKøberType` (translates into `hasKindOfBuyer`). After having done all this the model looks as shown in Figure 5.

4.2 Rule modeling - step II

Having added all the necessary classes and attributes to the model we are ready to model the rule itself. Since the rule describes a situation where you do not have to pay VAT we model it as a subclass of `Momsfritaget`. Following our naming convention we name the class `MomsfritagetSalgTilAmbassaderOgInternationaleOrganisationerIEU` (translates into `VATFreeSalesToEmbassiesAndInternationalOrganizationsInEU`). Based on the necessary and sufficient conditions captured in Table 4 we add the following necessary and sufficient conditions to `MomsfritagetSalgTilAmbassaderOgInternationaleOrganisationerIEU`:

- `harLeveranceType some Salg`
- `Varer` $\sqcup$ `Transport`
- `harLeveranceSted some EU`
- `harKøberType some AmbassadeOgPersonaleMedDiplomatiskeRettigheder`

The result is shown in Figure 6.

5 VAT Exemption 3: Sales in other EU countries

In this section we consider one final rule, the rule in Table 5. We identify the necessary and sufficient conditions for application of the rule. These are shown in Table 6.

5.1 Rule modeling - step I

We are already able to model that the rule concerns sale of goods delivered inside the European Union. The new thing is that we must be able to indicate whether a buyer is registered for VAT and if so, we must register the buyers VAT registration number. We use a functional data type property named `erKøberMomsregistreret` (translates into `isTheBuyerRegisteredForVAT`) with the data type `xsd:boolean` as its range to model whether the buyer is registered for VAT. Similarly we use a functional data type property named `erKøbersMomsnummer` (translates into `isBuyersVATRegistrationNumber`) with the data type `xsd:string` as its range to register the buyers VAT registration number if he has one.

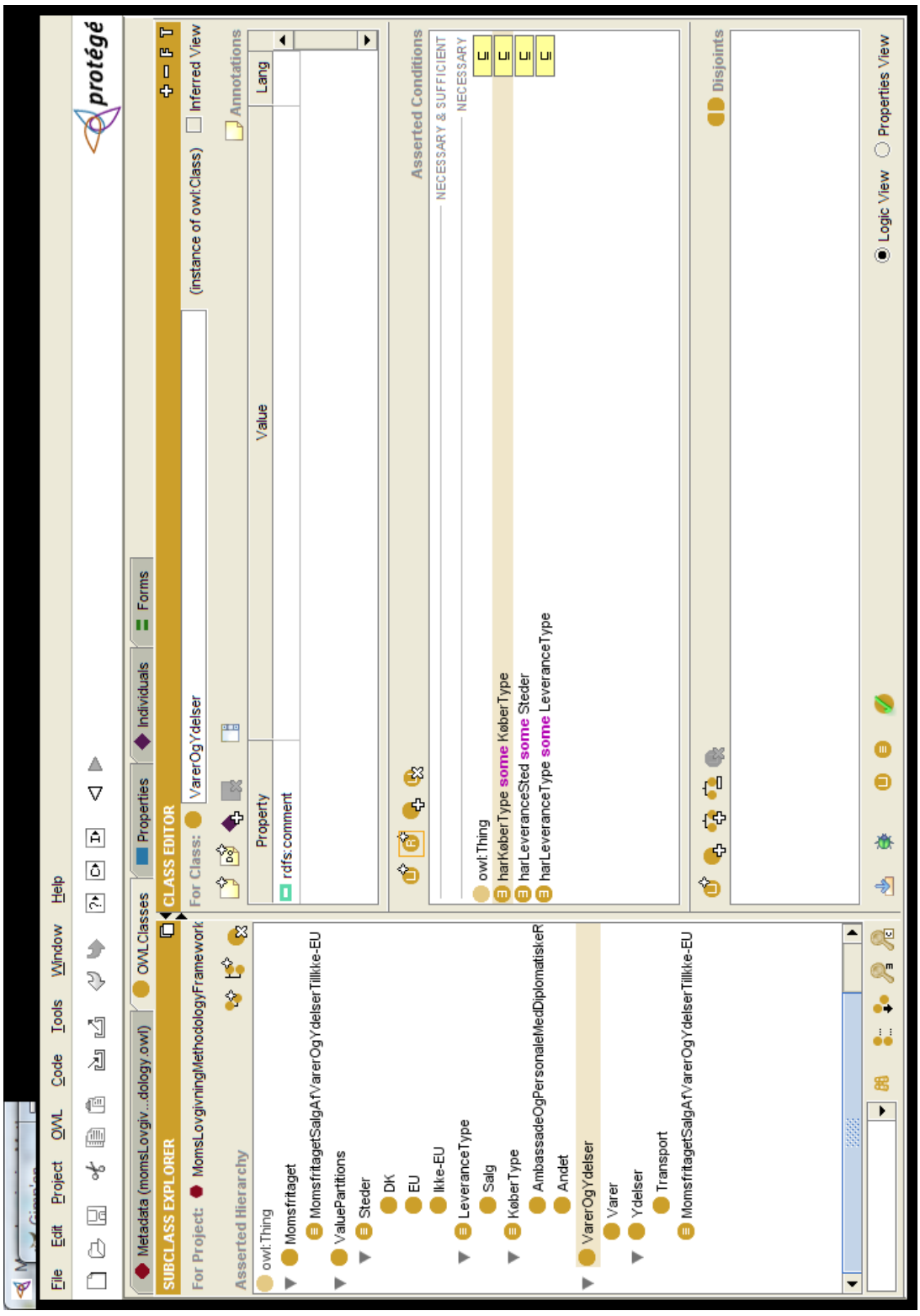

Figure 5: The model after adding classes and attributes as described in Section 4.1.

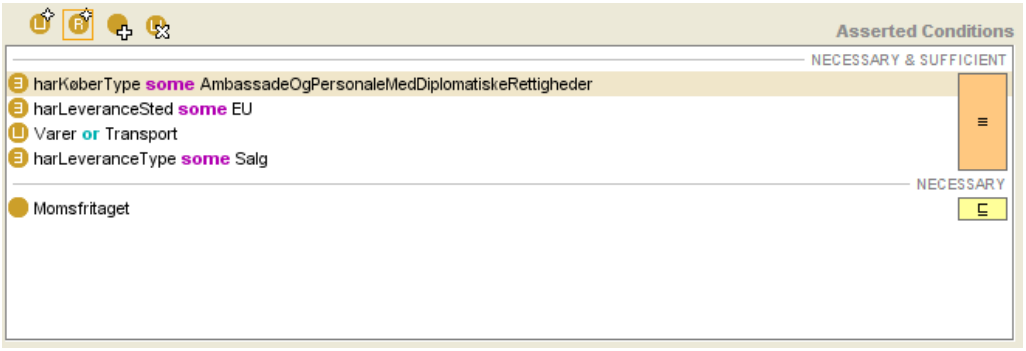

Figure 6: Asserted Conditions of our model of the legal rule in Table 3.

Table 5 Extract from the legal source and its translation into English.

> *Salg til andre EU-lande. Du skal ikke beregne dansk moms, når du sælger varer til momsregistrerede virksomheder i andre EU-lande. Du skal derfor sørge for at få virksomhedens momsnummer.*
>
> [4][p. 8]

And translated into English:

> *Sales in other EU countries. No VAT should be added to goods delivered to companies in other EU countries, provided that the companies are registered for VAT. In this case you must acquire the VAT registration number of the company.*
>
> Translated from [4][p. 8]

Table 6 Necessary & Sufficient conditions for application of the rule in Table 5.

- The rule concerns sales.
- The rule concerns goods.
- The place of delivery must be in the European Union.
- The buyer must be registered for VAT.
- You must acquire the VAT registration number of the company.

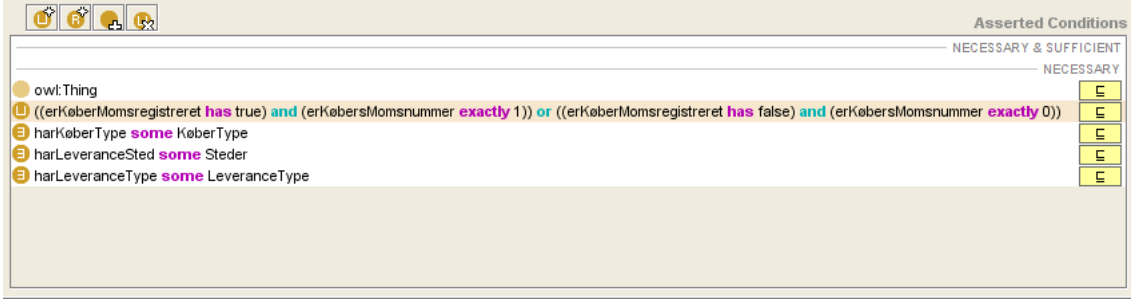

Figure 7: Asserted Conditions of `VarerOgYdelser` after adding the requirement for registering VAT registration numbers.

A read through of [4] will reveal that you must register the VAT registration number of the buyer exactly when the buyer is registered for VAT. Thus we model this as a property of `VarerOgYdelser` and not of `Varer` (as indicated by the rule). The requirement can be modeled as follows:

- `((erKøberMomsregistreret has true)` $\sqcap$ `(erKøbersMomsnummer exactly 1))` $\sqcup$ `((erKøberMomsregistreret has false)` $\sqcap$ `(erKøbersMomsnummer exactly 0))`

The result is shown in Figure 7.

5.2 Rule modeling - step II

Having added the necessary attributes to the model we are ready to model the rule itself. Since the rule describes a situation where you do not have to pay VAT we model it as a subclass of `Momsfritaget`. Following our naming convention we name the class `MomsfritagetSalgTilAndreEU-lande` (translates into `VATFreeSalesToOtherEUCountries`). Based on the necessary and sufficient conditions captured in Table 6 we add the following necessary and sufficient conditions to `MomsfritagetSalgTilAndreEU-lande`:

- `harLeveranceType some Salg`
- `Varer`
- `harLeveranceSted some EU`
- `erKøberMomsregistreret has true`

We note that the obligation to register the buyers VAT registration number is modeled indirectly, see Section 5.1. The result is shown in Figure 8.

6 Future work

Since this is work in progress there are a lot of areas we need to address. In the near future we plan to integrate our model in a prototype ERP system as described in the introduction. This opens the posibility for modeling the parts of the Danish VAT legislation concerning depreciation and VAT reporting (since they are intertwined and contain a lot of technical

Figure 8: Asserted Conditions of our model of the legal rule in Table 5.

requirements on the financial reports). We also need to model other countries VAT rules in order to confirm that Danish VAT rules are indeed representative with respect to the constructions that are needed in the modeling language. Based on this we need to refine our overall framework such that it captures the common structure and we need to identify what kinds of questions a model must be able to answer. The synthesized knowledge from modeling the VAT rules of other countries should also result in a more detailed analysis of what we can and cannot model.

Based on all this we should design a minimal description logic extended with the needed functionality identified in the analysis just mentioned, such as predicates like $x < 100$ which are needed in some rules. We should also provide a reasoner for the logic together with an editor such that the above process can be repeated.

Finally in order to compare our OWL model with a different approach we want to make a model using Datalog, which is the de facto standard language used to express rules in deductive databases, of the rules we have formalized in OWL already. It would also be interesting to try a hybrid solution e.g. OWL plus a rule language like SWRL. This work is independent of the tasks mentioned above and can be carried out in parallel.

7 Conclusion

We have shown how to model a subset of Danish VAT rules concerning exemption from VAT using Protégé-OWL. First we created an overall framework for the VAT model with the property that legal rules and the concepts they involve can be modeled as subclasses of existing classes in the framework. This helps to ensure that related concepts are modeled in the same way and that a single concept is not modeled twice. The second step was an iterative process consisting of two steps repeated for each rule. The first step is to extend the model such that the rule in question can be modeled. This is done by modeling concepts from the legal source as classes in the model and by adding attributes to the necessary conditions of such classes. The second step is to model the rule itself. This is done by adding specific requirements for application of the rule to the necessary and sufficient conditions of the class modeling the rule.

The step by step iterative modeling has been working fine in practice and an extension to cover several different VAT and duty rates does not seem to be problematic as long as they do not require us to model restrictions such as $x < 100$ which is not supported directly in OWL [7].

[7] Whether this is a weakness of OWL, or just us trying to use OWL for something it was not designed to

Apart from modeling inequalities we have not had modeling problems. One problem though is that reasoning about individuals in OWL models is not supported very well. Therefore we have tried to avoid the use of individuals wherever possible (using value partitions).

do is not clear at the moment.

References

[1] T.J.M. Bench-Capon, F.C.: Isomorphism and legal knowledge based systems. Artificial Intelligence and Law **1**(1) (1992) 65–86

[2] M. Horridge, H. Knublauch, A.R.R.S., Wroe, C.: A practical guide to building owl ontologies using the protégé-owl plugin and co-ode tools edition 1.0. (2004)

[3] Horrocks, I., Patel-Schneider, P.F.: Reducing owl entailment to description logic satisfiability. Lecture Notes in Computer Science **2870/2003** (2003) 17–29

[4] ToldSkat: Moms - fakturering, regnskab mv. (Vejledning E nr. 27) Version 5.2. (2005)