Detecting Contradictory Beliefs About Complex Systems Using Ontologies

Olivier Bellemans

University of Antwerp, Belgium
olivier.bellemans@student.uantwerpen.be

Abstract

Complex systems consist of large numbers of diverse, interacting components. The correctness of any one component rests on a set of assumptions about the rest of the system. Assumptions that are initially correct may become invalid as requirements change.

This project explores the use of ontologies to capture the different beliefs held about a train control system and to automatically detect inconsistencies in these beliefs.

Keywords: Ontologies, Description Logics, OWL, Consistency

1 Introduction

Complex systems consist of large numbers of diverse, interacting components. These components may be designed by experts from many disparate fields, using multiple formalisms. The correct behavior of a component rests on a set of assumptions made about the rest of the system. A software developer might assume a certain amount of processing power and available working memory from the hardware, for example. Assumptions that were initially correct may become invalid as system requirements change and evolve.

For this project, I've explored the use of ontologies to capture the different beliefs that are held about a system and to detect when multiple beliefs contradict each other. Specifically, I built an ontology for a train control system using the OWL Web Ontology Language[1] and the protégé ontology editor tool[2]. I used the integrated reasoning capabilities of protégé to automatically detect and pinpoint contradictory beliefs.

There's a brief introduction to ontologies in Section 2, followed by a description of the train control system and some of its requirements in Section 3. Section 4 shows how assumptions can be encoded in the ontology in a way that will lead to an inconsistent ontology if some of the assumptions are invalid. Finally, conclusions are drawn in Section 5.

2 Ontologies

Ontologies are descriptions of the concepts and relationships in a particular domain. Such a description may take the form of informal prose, or it may be embedded in a more formal language with precise semantics. In the latter case, we can use automated reasoning systems to derive new information that was not explicitly stated, or to detect contradictory statements.

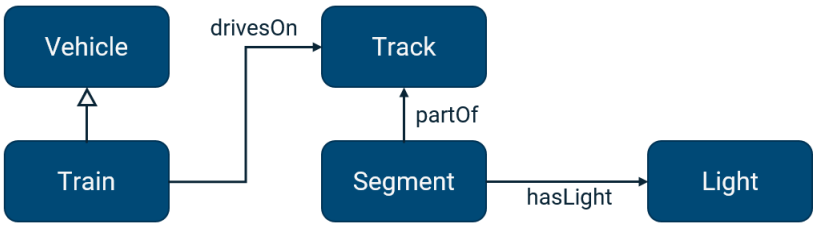

Figure 1: A small ontology capturing some essential concepts and relations of a railroad network. The empty arrowhead indicates Train is a subclass of Vehicle.

[1] https://www.w3.org/OWL/
[2] http://protege.stanford.edu/

The semantics of ontology languages are often based on a family of formal logics known as description logics [1]. There are many variants of description logic, each making a different trade-off between expressivity and efficiency of reasoning. Both major versions of the OWL Web Ontology Language are based on description logics. Each version supports several *profiles*, subsets of the language suited for different tasks. Each profile is based on a different variant of description logic. I mainly stuck to OWL1's DL profile for this project, it provides a good amount of expressivity while still having a decidable logic. For convenience, I did use OWL2's qualified cardinality restrictions, which lets you express concepts like 'a person with exactly two children that are male'.

In OWL, you can make statements both at the conceptual level ('trains are vehicles') and at the level of the individual ('Bob is a driver'). In this project I only make use of statements about concepts. The most important statement is saying a concept is *subsumed* by another concept (i.e. it is a subclass). If two concepts subsume each other, they are *equivalent*. Not all concepts need to be explicitly named. Concepts can be constructed and combined in various ways. You can take the intersection, union or complement of a concept. You can also construct a concept consisting of all individuals that have a certain relationship.

OWL defines a standard, XML-based format for describing ontologies. Since it can be a bit verbose and difficult to follow, I will convey ontologies using a visual notation loosely based on UML Class Diagrams instead. I will be explicit about the meaning of any new notation introduced.

3 Designing a Complex System

Complex systems consist of many diverse, interacting components. Design may involve experts from various fields working on heterogeneous components that ultimately require integration. Incorrect assumptions made in the design phase can cause integration to fail, leading to lost time and money.

In this section I describe a train control system, a complex system responsible for transmitting signal information and controlling and protecting trains on a railroad network. I sketch some requirements of this system and describe various plausible assumptions that could be made when designing its components.

3.1 System Overview

A schematic illustrating some key components of a train control system is shown in Figure 2. It is of the European Train Control System (ETCS)[3], designed to replace many of the incompatible European systems currently in use.

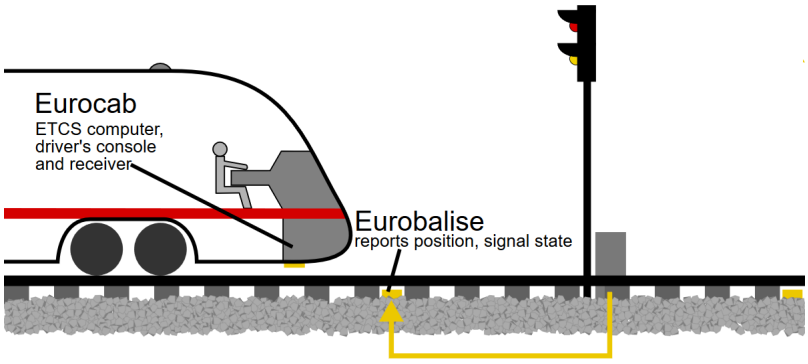

Figure 2: Schematic[4] of the European Train Control System. Placed between the rails is a transmission device that transmits location and signal information to the train when it passes over the device.

Placed between the rails is a *balise*, a transmission device that communicates location and signal information to the train when it passes over the device. On the bottom of the train is a receiver, which stores the received information in a computer system. This information is used to inform the

[3]http://www.ertms.net/

[4]adapted from https://en.wikipedia.org/wiki/European_Train_Control_System

driver, who might be driving at a velocity high enough that the external signals are difficult to discern, or to initiate certain actions automatically, like stopping the train in case of danger.

3.2 Making Assumptions

As stated previously, a requirement of the system is that location and signal information is communicated to the train when it passes over the transmission device. Among many other things, someone will need to decide on the layout of the digital data-frame that contains the relevant information. Since the train is in contact with the transmission device for only a brief moment, it should be verified that the data-frame is small enough to be sent in its entirety during that period. This depends on the data-rate of the transmission device and the maximum velocity of the trains. The person designing the data-frame will need to assume that these values stay within certain bounds.

Another requirement may be that the driver is warned when driving past a yellow light, and needs to press an acknowledgement button or the train is automatically stopped. A yellow light indicates that the next light is red and that the driver should prepare to stop at the next light. Here, it should be verified that when the automatic brake is applied, the train can actually come to a complete stop before reaching the red light. Otherwise, collisions can happen. Verifying this property might be done by a control engineer using a formalism like causal block diagrams or a domain specific modeling language to simulate the physical system. Whether a train can brake in time will depend on its weight, its maximum velocity and the distance between traffic lights. Again, assumptions need to be made about the system. When any of the assumptions are no longer valid, neither are any of the previous conclusions.

4 Capturing Assumptions

This section describes how assumptions can be incorporated into an ontology and how an automated reasoning system can detect inconsistencies between these assumptions and other beliefs. The ontology that we'll start from is shown in Figure 3. It contains the various families of trains that will be supported by the train control system.

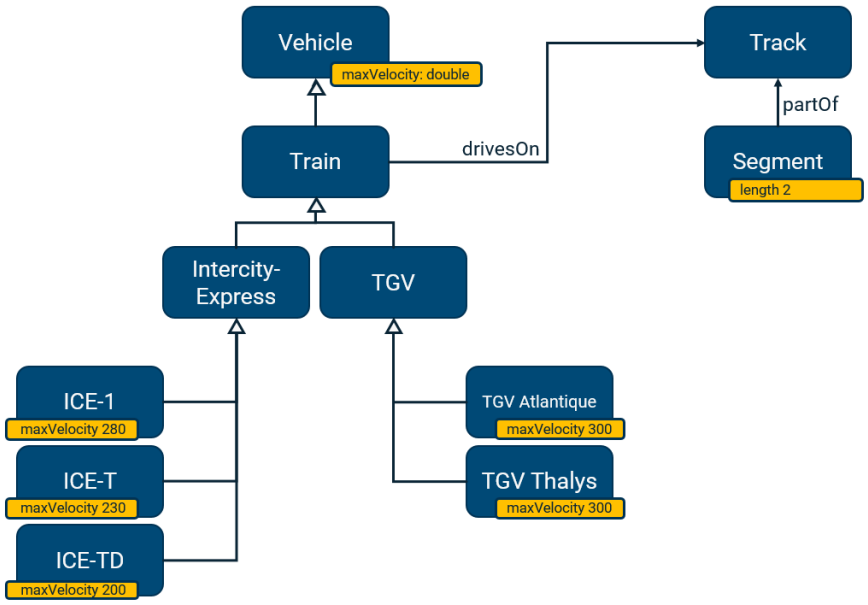

Figure 3: An ontology containing the different families of trains that need to be supported. Instances of each train family are constrained to have a certain maximum velocity, in km/h.

All vehicles, and thus all trains, have a property maxVelocity of type double. OWL has several builtin datatypes/concepts, one of which is the double precision floating point number. The exact maximum velocity is specified for all train families. Note that families like ICE-1 are not entities. They are concepts and we are stating that all instances of these concepts have the given maximum velocity.

4.1 Consistency

An inconsistency or contradiction occurs when two or more assertions are made that are in conflict with each other, i.e. can never be true at the same time. When you make two conflicting statements about a concept in an ontology, the result will be that there can never be a satisfying instance of that concept. Having an empty concept is not useful and a clear indicator of contradictory beliefs. Protégé's integrated reasoners can detect this kind of inconsistency and highlight the offending concept(s).

4.2 Encoding Assumptions

Returning to the earlier requirement of automatically stopping the train when a yellow light is ignored, one of the things that needs to be verified is whether the train can come to a complete stop before it reaches the next light. This depends, among other things, on the maximum velocity of the trains. The control engineer that verifies this property assumes that these values fall within certain bounds. In other words, he or she has a set of trains in mind for which the automatic brake system will perform as desired. As long as the trains that actually drive on the tracks are a subset of these, all conclusions drawn by the engineer will hold.

The OWL ontology language allows for concept descriptions such as 'all things with a maximum velocity of at most 300km/h'. A natural way to encode the assumption then, is to state that Train should be a subclass of this concept. One way to look at the subclass relation is as specifying a *necessary* condition. For something to be a member of a class, it is necessary that it has all the characteristics of the things in the superclass. (Similarly, the equivalence relation can be seen as expressing a *necessary* and *sufficient* condition).

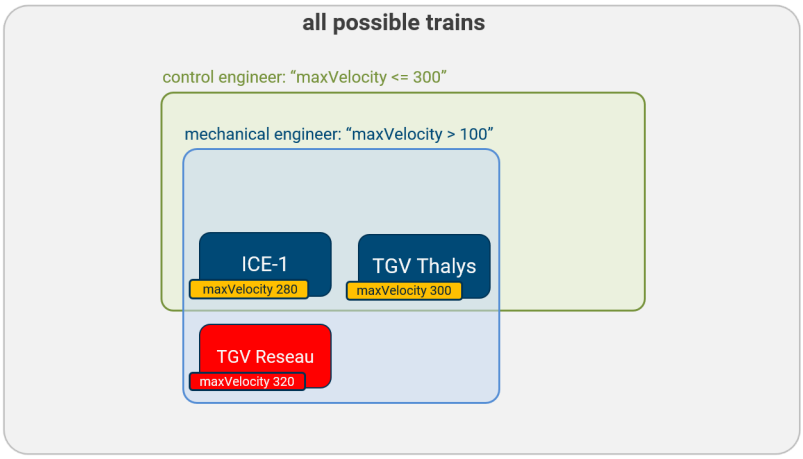

Figure 4: Two different assumptions placed on trains. Any family of trains must find itself in the intersection.

It is likely that other people will make different assumptions about the trains' characteristics. For example, someone may assume that all trains have a maximum velocity of at least 100km/h, as illustrated in Figure 4. Adding assumptions constrains the related concept (i.e. its set of possible instances grows smaller). The concept TGV-Reseau is overconstrained, as its instances should all have a maximum velocity of 320km/h, while simultaneously having a maximum velocity of less than 300km/h.

4.3 Introducing Inconsistencies

Adding assumptions about the train's maximum velocity and the length of track segments leads to the ontology in Figure 5. Here, the boxes with thick dashed borders denote unnamed concepts of all things whose properties with given name have values that fall in the given bounds.

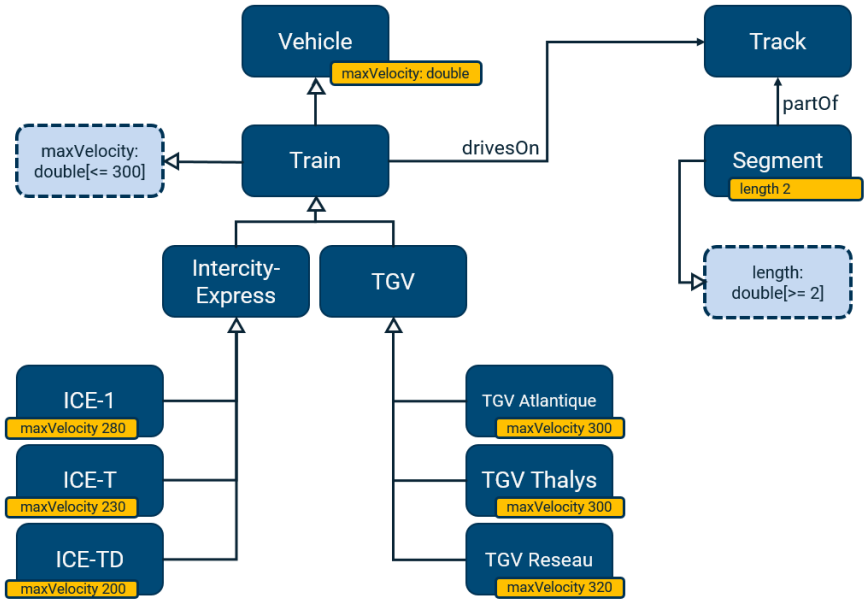

Figure 5: The assumptions made by the control engineer are encoded as unnamed superclasses. They are the boxes with the thick dashed borders.

Although the ontology was initially consistent, it was decided that the TGV-Reseau is to be supported. These trains have a maximum velocity of 320km/h, well above the velocity that our control engineer had assumed the trains would be operating at. Because we have stated that the TGV-Reseau is a Train, then by transitivity of the subclass relationship we also have that it is a subclass of the unnamed concept of all things with a maximum velocity of at most 300km/h. As a result, there can never by any instances of the TGV-Reseau concept, and it is clear that contradictory beliefs about the system exist.

Running protégé's integrated reasoner reveals that there is indeed an inconsistency and that TGV-Reseau is equivalent to the empty concept.

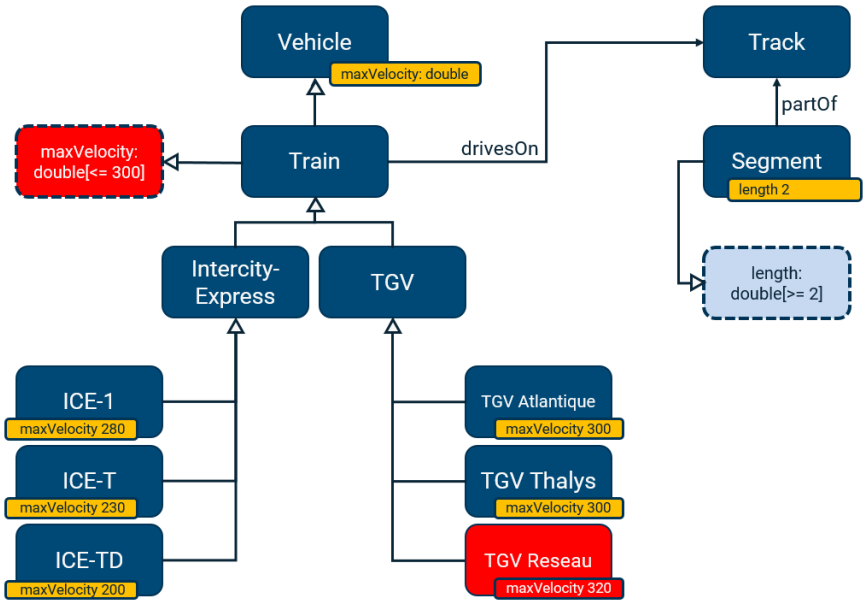

Figure 6: The beliefs that all trains have a velocity of at most 300km/h and that trains exist with a maximum velocity of 320km/h are inconsistent.

4.4 Pitfalls

OWL's semantics can be surprising at times. This mainly results from two key principles in its design [2]. One of these is that OWL does not make the closed world assumption. Under the closed world assumption, anything not known to be true must be false. Instead, if we were to add a train instance to the ontology without specifying a maximum velocity, there would be no complaints, even though we stated that all trains should have one. This is because under the open world assumption, it is assumed that this velocity exists, but is simply unknown to us.

Another cause for surprise is that OWL does not make the unique names assumption. That is, concepts or individuals with a different name are not assumed to be distinct (although they can be specifically marked as such). If we were to have a relation with maximum cardinality of one to two individuals with different names, rather than complain about the cardinality restriction,

the system will instead infer that these two individuals must be identical.

When you consider that OWL is a key technology for the vision of The Semantic Web [3], it is clear why these assumptions make sense. A concept such as Vehicle will certainly occur in many different ontologies. But due to the use of namespaces, the name will be different. Because there is no unique names assumption, you can indicate that your concept of Vehicle is equivalent to the Vehicle concept in a different ontology, which may contain much more information on vehicles than yours.

5 Conclusion

This project has explored the use of ontologies to capture the different beliefs held about complex systems and to reason about their consistency. I've given several examples of assumptions which may be correct initially, but become invalid once the requirements of the system change.

Using the OWL Web Ontology Language and the tooling that has sprung up around it, I was able to build a small ontology related to a train control system and encode the assumptions made by an engineer verifying a safety property. Once the assumptions were no longer valid, I was able to detect this using protégé's integrated reasoning capabilities, and pinpoint which concepts were the cause of the conflict.

References

[1] F. van Harmelen, F. van Harmelen, V. Lifschitz, and B. Porter, *Handbook of Knowledge Representation* (Elsevier Science, San Diego, USA, 2007).

[2] J. Heflin, *An Introduction to the OWL Web Ontology Language* .

[3] S. Grimm, A. Abecker, J. Völker, and R. Studer, Handbook of Semantic Web Technologies , 509 (2011), arXiv:1011.1669v3.

[4] K. Vanherpen *et al.*, 2016 1st International Workshop on Cyber-Physical Production Systems, CPPS 2016 (2016).

[5] B. C. Grau *et al.*, Web Semantics: Science, Services and Agents on the World Wide Web **6**, 309 (2008).

[6] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning About Systems* (Cambridge University Press, New York, NY, USA, 2004).