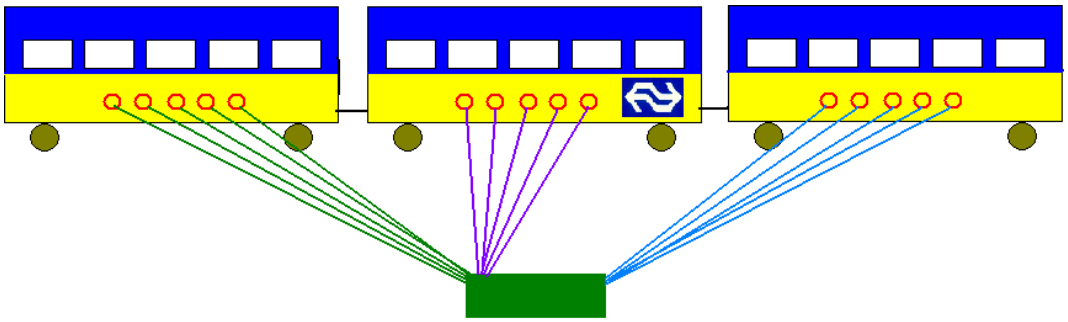
MSc. Thesis

An ontology-based Bayesian network approach to aggression analysis in a train environment

by
Vishal Gangaram Panday(1015117)



March 2008

Faculty of Information Technology and Systems
Man-Machine Interaction Group

Acknowledgement

I want to take this opportunity to express my gratitude to all the people for their contribution and support they gave me to accomplish this thesis. The work presented in this thesis is my own, but I could not have accomplished this without the help of others. It has been a long journey and some people around me would say too long. Nevertheless the time has come to express my gratitude to the following people for their contribution and support because I could not have done it alone.

Firstly, I would like to thank my supervisor, Drs. Dr. L. J. M. Rothkrantz, for the opportunity and his guidance to bring this thesis to a good end. I also would like to thank Ir. Z. Yang for all his inspiration, feedback, guidance and patience with me and not to forget the hours in the weekends to brainstorm on different research topics. His knowledge and instructions helped me to conquer a lot of challenges in the process of my research.

Furthermore, I would like to thank both my lovely sisters for supporting me in difficult times. Last, but certainly not least, I would like to thank my parents, Ramanand and Marlene Gangaram-Panday in Suriname, who supported me financially, emotionally and mentally and also gave me the opportunity to make this possible. Thank you everyone from the bottom of my heart for this opportunity and your warming love and belief in me.

An ontology-based Bayesian network approach to aggression analysis in a train environment

by
Vishal Gangaram Panday(1015117)

Submitted for the degree of Master of Science in
Media and Knowledge Engineering

March 2008

Delft University of Technology
Faculty of Electrical Engineering,
Mathematics and Computer Science
Man-Machine Interaction Group
Mekelweg 4, 2628 CD Delft

Graduation Committee:
Drs. Dr. L. J. M. Rothkrantz
Dr. Ir. C.A.P.G. van der Mast
Dr. K. van der Meer
Ir. Z. Yang

Abstract

Aggression and violence can be used in a very broad way in our daily life. If you pick up a newspaper or watch the evening news, there is always some instance of aggression. The impact of any form of aggression can vary from simple pain to great depression. To cope with aggression detection in an automated way there are some technical challenges that can vary from human and group tracking, speech/audio recognition, facial recognition to behavior recognition. The main challenge for this thesis is to put all the objects and concepts in our domain in the right context of each other. A first step is to model the train environment domain. To cope with the task of detecting aggression in trains the Dutch Railways (Nederlandse Spoorwegen, NS) and the Man-Machine Interaction Group at the Delft University of Technology are working together on this project.

The main challenge for this thesis is to put the main objects and concepts in our domain in the right context of each other. An information system that is capable of detecting aggression must first be aware of the objects in the environment, their capabilities, and their relationships and perform automated semantic interpretation of events in the environment. This can be done by designing an aggression ontology, which is the first task. Ontologies are computer-stored specifications of concepts, properties, and relationships that are important for describing an area of expertise(domain).

The next task is to design a Bayesian Network(BN). The choice for a Bayesian approach was such that we had to find an appropriate tool able to represent various sources of uncertain information that describe our problem, and to join them into an inference system. Our proposed Bayesian network will handle incoming data from an annotating process using the concepts from the ontology. The output will give a probability for the 5 aggression scenario classification (neutral, damage, annoyance, danger and sickness). A case study on how real data(acquired from actual footage shot in a train with actors), can be modeled in our Bayesian Network and inferred with will be described and evaluated in this thesis.

Keywords: Ontology, Bayesian Network, Aggression detection.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Human aggression detection has received attention from computer vision researchers in recent years. This interest is motivated by a large number of real world applications including video surveillance in train compartments in which a constant routine observation of the scenes by human operators is required. Automatic (or semi-automatic) activity recognition for video surveillance applications can relieve such tediousness or improve its efficiency. Security people were always interested in a reliable and robust automated surveillance system to take the place of the human operators monitoring the scene. The task in automatic activity recognition for video surveillance applications is largely composed of 1) detecting and tracking the moving regions, and 2) recognizing the type of mobile objects and their activities. To cope with the task of detecting aggression in trains the Dutch Railways (Nederlandse Spoorwegen, NS) and the Man-Machine Interaction Group at the Delft University of Technology are working together on this project.

Aggressive and violent behavior by one human being toward another is not a new phenomenon. In train compartments some forms of aggression can be considered typical. Passengers can be harassed, robbed, intimidated and/or hurt in any other way. Aggressiveness and problematic behavior in public places can cause great distress on the part of innocent bystanders, which can often lead to mental and physical stress and even destruction of property. Where as different forms of aggression are trivial to detect by humans, it's not for computers. An automatic system will need to recognize speech, behavior, body language as well as potentially harmful objects. The system will also need to be able to track people and groups of people traveling together.

The task for this thesis is a step towards an automated aggression detection system. First we have to design an aggression ontology which will describe the domain in detail. Ontologies are computer-stored specifications of concepts, properties, and relationships that are important for describing an area of expertise(domain). The domain in this case is a train compartment which contain concepts(entities) that can be related to each other via properties. A detailed de-

scription of ontologies will be presented in a later chapter in this thesis. Next this aggression ontology will be used as a basis to design a Bayesian Network(BN) which will be capturing the uncertainty relations between the concepts in the domain with the goal, given the relations and probability distributions, to predict the scenario in the train compartment(probability inference). A Bayesian Network (BN) is a directed acyclic graph and we will see in chapter 6 fits the basic structure of ontologies very well. Vertices(nodes)in the BN are considered as concepts, and labeled edges as relations.

1.1 Relevance

The Dutch Railways(NS) are spending millions of euro's every year on safety and security of their passengers in trains and stations. From the annual report of 2006 [1] it can be seen(figure 1.1) that the percentage of customers giving a mark 7 or higher for safety has increased to 74%. And from the 2007 half year report that amount has increased to 76%. Not the least to note that the number of commuters using the trains has increased with 5% in 2006 while still maintaining good security in and around the trains. To further increase safety different innovative methods, like the project with the Delft University of Technology to design an intelligent system to detect aggression are tested.

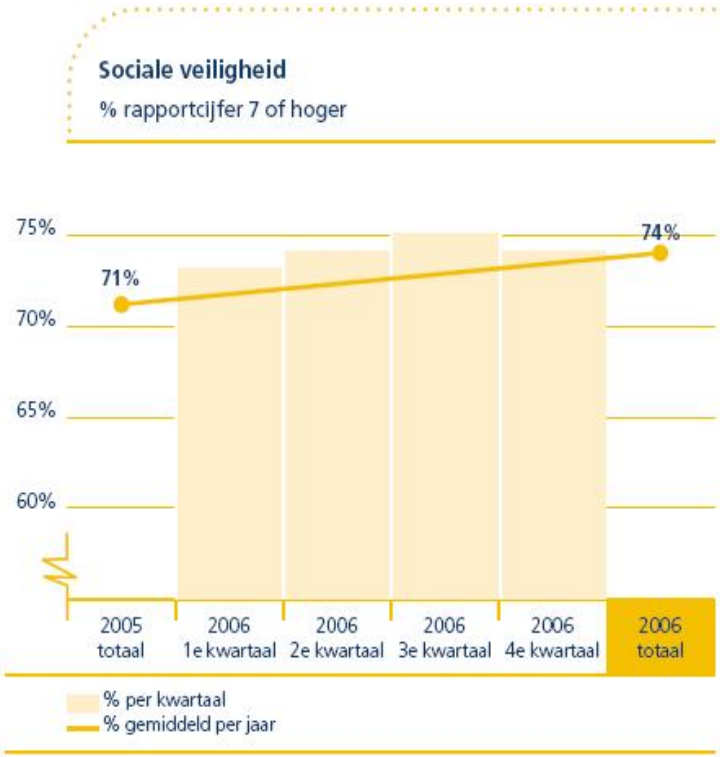


Figure 1.1: Safety mark given by customers[1].

Such a system has several advantages in both social and economical ways. The bombings on metro and train station in London and Madrid in the past years,

made the need for such a surveillance system take flight. This was also necessary for the commuters to feel safe again when going to work or school using public transport. Sensors generate a lot of data, so to monitor for example all train camera's in the country will require a lot of manpower which is expensive. With an automated system this will be significantly cheaper to achieve.

To be able to detect aggression in an automated way and favorably real-time we can spare commuters a lot of pain and grief. To cope with the security of people we see more projects on aggression detecting starting up. Some only use single modalities, like sound or video, but the project at the Delft University of Technology wants to investigate the fusion between video and audio data and analyze the features that contribute to aggression.

1.2 Problem Definition

Aggression and violence can be used in a very broad way in our daily life. If you pick up a newspaper or watch the evening news, there is always some instance of aggression. The impact of any form of aggression can vary from simple pain to great depression. To cope with this, automated aggression detection has to overcome some technical challenges that can vary from human and group tracking, speech/audio recognition, facial recognition to behavior recognition. The main challenge for this thesis is to put all the objects and concepts in the problem domain in the right context of each other. The idea is that an information system that is capable of detecting aggression must first be aware of the objects in the environment, their capabilities, and their relationships and perform automated semantic interpretation of events in the environment. However some questions that needed to be covered first:

1. Which scenarios can occur in a train compartment?
2. Which of these scenarios can be considered as a threat of the safety in a train? And how can we quantify this?
3. Which concepts in a train compartment do we need to model?
4. What features of people in a train compartment are important?
5. How can we combine these features and concepts in a train compartment to evaluate the scenario?

1.3 Goals

The first goal for this thesis is to design an aggression ontology which will model the concepts in a train compartment and the interrelations between concepts. The question may rise why an aggression ontology is needed for

aggression detection. The answer to this question can be given in two ways. Firstly to detect aggression is it not enough to know "what happened" (e.g to detect situations in the train). Events of the objects in a train environment need to be placed in a proper context and in proper relation to each other. This implies that we need to know the causal relation and correlations about the objects in relation to how an event happened. If an event took place we need to know the impact it has(aggression). This is only possible if we reason about the concepts in the aggression domain. Thus it is important to design an aggression ontology. A second reason for an ontology is that we have collected data of aggressive situations in the train environment. For various reasons it is necessary to annotate the data to make the semantics explicit. However, different annotators might use different vocabularies to annotate. An ontology-based annotation approach, in which an aggression ontology is used so that the concepts and their relationships are formally defined will allow annotators to use the same vocabulary to annotate the data, and reasoning systems driven by the ontology will be able to process the data with greater precision. In other words, the process of creating recognition systems for high-level analysis of surveillance data can be largely automated, provided sufficient quantities of ground truth data which has been annotated with descriptors from the desired analysis specification are available. Such a specification may usefully be regarded as an ontology which provides a prior description of the application domain in terms of those entities, states, events and relationships which are deemed to be of interest. The hierarchical organization and relational constraints imposed by such an ontology may then be used to guide the design of a complete aggression analysis system.

This ontology will be modelled using the Ontology Web Language(OWL). The second goal is then to transform the designed ontology to create a Bayesian Network(BN) which will incorporate the different uncertainty levels of the concepts in the train. The BN could be used to compare the conclusion of sample data with the reality. There is audio and video data available from actors performing different aggression scenarios in the train. An annotation tool was implemented by Ismail[32] to a help the annotation process and guide an annotator through the annotation process. During the annotation process the annotator can note the presence of (pre-defined) features, the occurrence of events, and the existence of relationship between entities. There is data(audio and video) available, where we had actors play different scenarios in a real train environment.The data from this footage, could be properly annotated using this tool and the aggression ontology designed in this thesis.

1.4 Approach

In this thesis we try to achieve a better understanding of a complex problem. We develop the understanding by analyzing the context of the problem and comparing it with problems and solutions in related areas. To start off, a literature survey was done on the background of aggression, what ontologies are, how they can be used and ontology design. This included techniques used in related projects. We also assess previous research done within the Man-Machine Interaction(MMI) group, including report of interviews with experts.

Secondly a questionnaire(see Appendix B) was done on commuters using the trains daily to get a idea of which type of situations that can occur in a train they feel are threatening and which are not so threatening or neutral. The usage of the resulting data could be handy to categorize the different situations in the train. This survey was done as to gain some extra information about how commuters perceive certain situations as they happen in a train. Question about situations that can occur in trains were asked with a oridnal scale from 1 to 5 on how they would rate the occurring situation.

Third phase was to design the aggression ontology. Our first step was to define a proposed ontology approach. As mentioned before, an ontology provides consistent and unambiguous data definitions and relationships. To launch our ontology development effort, we researched relevant ontology and OWL efforts to help us in our approach. Also we researched several methods and tools to design ontologies using the Ontology Web Language(OWL).

The last phase is to convert the designed aggression ontology into a Bayesian Network(BN). The choice for a Bayesian approach was that we had to find an appropriate tool able to represent various sources of uncertain information that describe our problem, and to join them into an inference system. This tool should be able to represent a degree of uncertainty, i.e. the probability of the objects or events, and relations that exists between these objects, and events. In probability reasoning, random variables are used to represent event and/or objects in the world. Bayesian Networks bring the most appropriate representation of relative influences among real world facts. A case study will also be done on how real data(acquired from actual footage shot in a train with actors), can be modeled in our Bayesian Network.

1.5 Thesis Structure

The first chapter mainly gives an introduction to the work done for this thesis and the relevance for the work in general. Also the problem definition is given followed by the goals and the approach. In **Chapter 2** we will give a brief background information about the work done in the literature survey about the human aggression and the different types of aggressions known from literature.

In **Chapter 3** we will give a broad introduction of what ontologies are, how they can be used, tools that are available as well as the different ontology languages that are available. Discussion about ontology reasoning in general will be covered also in this chapter. In **Chapter 4** we will discuss the design of the aggression ontology as well as the concepts, properties and relationships. **Chapter 5** will present the necessary background theory about Bayesian Networks(BN) in order to understand the design of the BN presented in chapter 6. Also an overview of tools used to achieve this will be covered. In **Chapter 6** a Bayesian Network will be designed with the aggression ontology as the basis. **Chapter 7** will be a case study done with actual data to illustrate how our Bayesian Network can be used on it. **Chapter 8** will give a brief overview and concludes this thesis with recommendations that can be researched or developed further in future work on this project.

Chapter 2

Theory of Aggression

This chapter gives background information about the core of human aggression and the types of aggression known.

2.1 Human Aggression as a research topic

Nowadays it is virtually impossible to pick up a daily newspaper, browse through a popular magazine, or even tune in to the evening news without learning of the occurrence of some instance of aggression or violence. Although the majority of our interactions with other people do not involve aggression or violence, such behaviors are the source of a great deal of physical pain and psychological damage. In this chapter we will primarily focus on human aggression. Since aggression by men and women seems to involve many factors unique to human behavior (e.g., the desire for revenge, racial or ethnic prejudice etc.), it seems reasonable to concentrate on this topic. Secondly, discussion about aggression done from a social perspective. Aggression will be viewed as a form of social behavior involving direct or indirect interaction between persons. Aggression, in case of human beings, originates primarily from the words, deeds, presence, and even appearance of other persons [21]. The difference between physical and verbal aggression is obvious. A little less obvious is the distinction between direct and indirect aggression. Indirect aggression is committed outside the presence of the victim, where direct aggression is a face-to-face confrontation with the victim. In a study [5]about the gender that are involved in the forms of aggression, we learn that females are more likely to engage in indirect forms of aggression and males are more likely to engage in direct physical aggression. Also evidence from the study show that both genders are about equally likely to engage in verbal aggression. Example research groups that study aggressive behavior in humans are in the fields of psychology and biology. The psychological field tries to study the evolution of human aggression over time, while the biological field studies the genetic inheritance from ancestors. In the next sections we go deeper into different forms of aggression,

also from different perspectives(e.g biological, psychological) and discuss what can be learnt from the different fields.

2.2 Definition of Aggression

Few people would deny that aggression is a commonplace in the existing society we live in. For some, such as those living in the Middle East, parts of Africa and any maximum security prisons, aggression and violence are experienced daily and in intensely personal ways. For others the phenomenon is known mainly in indirect ways, such as through motion pictures and television. Aggression, whether painful to life or ego, seems to be a real and important part of the human condition. The broad way the word has been used, makes the study of aggression difficult. Does it make sense to use the word "aggression" to refer to not similar events such as fight during a basketball match, quarrel in the train or bombing of a public place? Its usefulness is limited by the numerous motives and forms of expression that characterizes aggressive behavior. In our research we have come across many definitions of 'aggression'. One general definition giving by Buss [8]: Aggression is 'a response that delivers noxious stimuli to another organism'. Many psychologists would agree with this definition, but it is admitted that this definition may not cover all examples and that it can be attacked on several points. There are many variables in aggressive behavior that this definition does not cover. Simple examples are the role of emotions in such aggressive actions or social-cognitive judgements that occur before such actions. These are some points which the basic definition of aggression can be attacked with. In another study, Berkowitz [2] pointed out that one of the problems in defining aggression is that in the English language the term is used to refer to a large variety of different actions.

> *When people describe someone as being aggressive, they might be saying that he frequently attempts to hurt others, or that he is often unfriendly or, in a quite different sense that he is typically very forceful and tries to get his own way in this dealings with others, or maybe that he is assertively stands up for his beliefs, or perhaps that he usually attempts to solve the problems facing him.*

This definition includes several key elements which are:

- Aggression is a behavior, not an attitude, motive, or emotion;
- An intention exists to cause harm to the victim;
- Some types of aversive consequences occur;
- The victim is a living being;

- The victim is motivated to avoid harm.

Thus, in studying human aggressive behavior we are faced with the following issue-how to define the basic concept in a meaningful and useful way. This will be discussed in the following sections.

2.3 Types of Aggression

Attempts to understand human aggression at a generic level, as actions intended to hurt someone, have been criticized as inadequate unless researchers distinguish between different types of aggression and their distinctive determinants and regulatory mechanisms [9]. Most people, when they hear the word 'aggression' they immediately think of the physical force - a harsh verbal harassment, a fight, an attack with a weapon, or any other form of intense action in the conflict between two people. According to the definition discussed above aggression can be carried out in any behavior by an intent to harm another person. Behaviors like damaging or destroying one's property or spreading nasty rumors about someone in order to destroy that person's reputation can also be seen as effective ways of aggressing against that person. These are examples of indirect aggression where there isn't a face-to-face confrontation between the aggressor and the victim.

Discussions about aggression draw a series of divided distinctions between types of aggression. The main pairs are affective versus instrumental, impulsive versus premeditated, and proactive versus reactive. These pairs are typically conceived in overlapping ways, leading to some confusion. Affective aggression [18], or emotional [3] is usually conceived as impulsive, so thoughtless (that is, unplanned), driven by anger, having the ultimately motive of harming the target, occurring in reaction to some perceived provocation. Instrumental aggression, in contrast, is usually conceived as a premeditated means of obtaining some goal other then harming the victim, being proactive rather than reactive, and resulting from cold calculation rather than hot affect. Impulsive aggression is usually conceived as thoughtless (automatic, fast, and without consideration of consequences), reactive, and affect laden. Premeditated aggression, in contrast, is usually conceived as thoughtful (deliberative, slow, and instrumental), proactive, affect-less. Proactive and reactive aggression are frequently used interchangeable with instrumental and affective, but with slightly different emphases. Proactive aggression is usually conceived as occurring without provocation, is thoughtful, and has little or no affect. Reactive aggression is a response to a prior provocation and usually is accompanied by anger [16] [38].

2.3.1 Instrumental Aggression

The two important characteristics of instrumental aggression are goal-directness and planning. The instrumental aggressor acts to obtain a readily apparent goal for example money. Examples of instrumental aggression include shooting a police officer in the course of a bank robbery, stabbing a homeowner during a burglary, and strangling a rape victim. Rape is almost always instrumental. Instrumental aggression is initiated as a means to an end rather than as an act of retaliation or self-defense.

Instrumental aggression, most of the times, are preceded by good planning or preparation. However, in some cases instrumental aggression involves relatively little planning, such as in the case of a criminal who engages in an opportunistic offense (e.g., unexpected opportunity to rob someone that involves assaulting the victim). In some cases, a subject may plan a robbery or burglary, and when something goes wrong, engages in an act of aggression, such as shooting someone in order to get away. Here in these cases it should be considered that the subject's plans include the possibility of violence, even if there was no specific plan to shoot someone.

Instrumental aggression usually involves little or no provocation by the victim. In some cases subjects may be "provoked" into violence in the course of another crime, e.g., a robbery victim who insults the subject or resists the robbery in some way. These acts are still considered instrumental acts of aggression.

Instrumental aggressors are motivated by goals, not emotions. It follows that their level of emotional arousal, especially anger, is relatively low or is secondary to the act. Some instrumental aggressors try to calm themselves prior to an offense through drug use or drinking. In extreme cases, instrumental aggressors are not angry toward their victims and may have a cold, "business-like" attitude about their behavior. Nevertheless, many less hardened instrumental aggressors are nervous and highly aroused while committing a crime, even though it is not their arousal which motivates their actions.

The term "instrumental" should not be defined so broadly that it encompasses all aggressive behavior simply because there is a definable goal or desired outcome to the aggression, such as warding off an attacker or taking revenge on someone. Aggressive behavior whose purpose is to defend against a threat or in some way respond to provocation is defined as reactive/hostile aggression. If the subject is engaged in some form of criminal activity, such as a drug deal, associated violence is almost always instrumental.

2.3.2 Affective Aggression

Most of the times aggression is associated by strong negative emotional feelings such as anger. Anger is usually aroused by some provocation. Anger is most

often thought of as an intervening condition that instigates, and then guides, affective aggressive behavior aimed primarily at injuring the provoking person. It is accompanied by distinctive patterns of activity in the central and autonomic nervous systems [34]. The idea that a flash of anger can inspire retaliatory aggression is easy enough to grasp. Sometimes, however, retaliation comes so long after provocation that we find it difficult to attribute the action to anger, an emotion that is relatively short-lived for most people, even though the retaliatory aspect of the anger is still apparent.Frijda[43] has commented on the possible emotional state involved in this sort of delayed response affective aggression. It has many of the properties of anger: it is a state of impulse, it disposes the person to action, it is often accompanied by bodily arousal, it can become a preoccupation that takes attention away from other matters. But it differs from anger in other ways, one of which is the often extended duration between provocation and response. Frijda [43] suggests that this condition is not an emotion, but a complex cognitive state having close links to emotion. Such a state may be labeled a 'sentiment': the emotion of anger towards the other person becomes in time transformed into the sentiment of hatred, which outlives the original anger. Long-term feuds and grudges represent cases in which people are aggressed against because they are hated, not because they have done anything in particular to elicit anger in the aggressor.

2.3.3 Reactive Aggression

The two important characteristics of reactive/hostile aggression are reaction to provocation and arousal of hostility. Aggressive behavior represents reactive hostility to the extent that the aggressor reacts to perceived provocation or threat by the victim. The provocation may include insults, threats of aggression, or other acts that frustrate and anger the aggressor. The objective of the aggressive act is to harm or injure the victim, in response to feelings of hostility that may include a mixture of anger, resentment, fear, or other distress aroused by the victim's actions. Typically, there should be some form of interpersonal conflict (argument, dispute, prior aggression) between aggressor and victim. In many cases the aggressor and victim have a prior relationship as relatives or acquaintances, but in other cases there is no prior relationship and the parties are strangers to one another.

Reactive/hostile aggression can involve extended time-frames. For example, an abused family member may plan an ambush to rid the family of the abuser. The most recent episode of abuse could be long before the aggressive reaction. The critical issue is that the reactive/hostile subject is reacting to an interpersonal conflict that arouses hostility.

Many instrumental offenders may be angry at someone else, upset over a failed relationship, lost job, etc. This provides a context for understanding the person,

but it should not enter into the determination that a person engaged in instrumental versus reactive/hostile violence. A person who sets out to rob a bank is committing an instrumental act, regardless of any prior life stress. A person who is embroiled in an intense interpersonal conflict with the victim will commit a reactive/hostile violation.

2.4 Other Views of Human Aggression and Treatment

If we are already taking a look at aggression, it is also good to have a look at aggression from a biological and social point of view. Aggression is defined as the delivery of a noxious stimulus to another person with the intent of harming that person, in the expectation that the aversive stimulus will reach its destination, and without the consent of the victim. In humans, aggression in human beings takes one of two general forms: (a) angry, or affective; and (b) instrumental. Numerous explanations for affective aggression have been formulated. One, based on an evolutionary and biological viewpoint, is that humans share with other animals certain genetically determined tendencies towards aggressive behavior. Another, based on a behaviorist position, emphasizes the acquisition of aggression through experience, conditioning and learning. The two views are not mutually exclusive. Some aggressive behaviors in humans have biological origins, just as some are learned through observation of other people. Aggressive behavior, is shaped and developed through learning processes. Both genetic inheritance and learned tendencies serve as predisposing background conditions for aggression, which is a response to provoking conditions in the persons environment.

Several intervention programs have been developed over the years to treat overly aggressive individuals, but they have failed. Social-cognitive theories claim that treatment of such individuals become increasingly less successful as the individuals become older. A simplified explanation is that with increasing life experiences ones interpretations of the social world is based on increasingly well-rehearsed and accessible knowledge structures, which are very difficult to change. However, treatments can have beneficial impact on juvenile offenders with a tailored intervention program to fit the individual constellation of contributing factors [41]. Treatments that have failed on juvenile offenders such as boot camps, individual therapy, group therapy because they did not address the range of factors that contribute to the developments and maintenance of violent behavior. One approach that did work was the multisystematic therapy by Borduin [6]. This approach basically identified all major factors contributing to the delinquent and violent behaviors of the particular individual undergoing the treatment. Once these factors are categorized an intervention is then tailored to fit the group of contributing factors.

Brain mechanisms and the activity of hormones have also been implicated in human aggression as background factors. Elevated levels of testosterone have been found to co-vary with aggressiveness in males by creating a heightened disposition to aggress in response to suitable provocation. Some studies have also shown that situations involving competitive and assertive behavior can lead to elevated testosterone levels, suggesting that the relation between hormone activity and aggression may be in part a reciprocal one. Both the limbic system and the cerebral cortex are linked to aggression, the former as a primitive center of emotional reactivity to provocation and the latter as a higher center exercising cognitive controls over emotional responding. In particular, dysfunction in the frontal-lobe region of the cortex is correlated with aggressive behavior and mood. The activity of the neurotransmitter serotonin is involved in aggression, in that relatively low levels of serotonin activity such as may be induced through damage to the limbic system or inhibiting drugs are associated with high levels of aggression.

The research field of aggression and human behavior is so broad that it's impossible to cover all aspects of aggression in this thesis. It's also possible to go in depth about the emergence of aggression between genders. Also biological deficiencies that can cause aggressive behavior could have been discussed here. Think about ADHD (attention-deficit hyper-activity disorder) or low levels of the neurotransmitter serotonin 5-Hydroxytryptamine (5-HT) which are related to heightened aggressiveness [31]. We tried to keep it brief and focused on the types of aggression which are the most important aspects. Intervention and other types of biological or gender differences are beyond the goal of research but could be interesting. We think that the main (and probably the most important) conclusions to be gotten from this discussion (in relation to this project) is that emotions(affective aggression) and goals(instrumental aggression) can play an important part in the detection of aggression.

2.5 Aggression in the Train

It costs the Dutch Railways(NS) per year millions of euros to execute measures for prevention of aggression in their trains and especially known "trouble" routes. The first priority for the NS is to keep their conductor and other train personnel safe. Then the safety of its passengers comes next. In the train aggression can be towards the conductors in the form of swearing or fighting. Also passengers can physically or mentally hurt other passengers with their behavior. Another form of aggression is vandalism e.g. damaging the trains interior, by painting, writing or breaking the seats, windows, tables and doors. This form is very expensive for the NS. Also the misuse of the emergency brake can be considered a form of vandalism. If any of the above forms occur in a train at a given time then the train tables will be mixed up and delays are the

results which costs the railway companies again millions of euros.

2.5.1 Aggression categorisation by the Dutch Railways

The Dutch Railways (Nederlandse Spoorwegen (NS)) has an incident categorisation as defined in table 2.1.

Table 2.1: Incident categorisation used by the NS

Category	Description
A	Suspicious behavior
B	Robbery and theft
C	Violence
D	Serious public inconveniences
E	Small public inconveniences
F	Vandalism
G	Accident
H	Fire

We will make a selection of the most important categories of aggression into our ontology. For example serious and small public inconveniences we will cluster them into annoyances. We won't be taking Fire into consideration because that is out of our scope and for the Dutch Railways it belongs to an incident and not really to aggression. In chapter 4 we will discuss our categorisation as we see best fit to model aggression in the train compartment domain.

2.6 Related Research

This part will give a brief overview of partly correlated studies about situation awareness and a study about behavior recognition systems. This section is meant to show some application areas for behavior recognition. Although it is not the same as aggression detection, we can see some correlation between behavior and aggression.

2.6.1 Behavior Recognition Systems

In any system that will have some form of a behavior recognition module there will be some form of tracking module too, to follow certain movements of an actor in the scene. Such a system has been proposed by Cupillard et.al [13] where a behavior recognition module relies on a vision module composed of three tasks: (a) motion detection and frame to frame tracking, (b) multiple cameras combination and (c) long term tracking of individuals, groups of people and crowd evolving in the scene. For each tracked actor, the behavior recognition module performs three levels of reasoning: states, events and scenarios. The vision module is composed of three tasks. First a motion

detector and a frame to frame tracker generates a graph of mobile objects for each calibrated camera. Second, a combination mechanism is performed to combine the graphs computed for each camera into a global one. Third, this global graph is used for long term tracking of individuals, groups of people and crowd evolving in the scene (typically on hundreds of frames).

The motion detector and frame to frame tracker has 3 sub-tasks: detection of mobile objects, extraction of features, classification of mobile objects. A list of mobile objects is obtained at each frame. Once all mobile objects are extracted for each camera they are added to a graph. All the graphs for each camera with all mobile objects are now combined into a combined graph (see figure 2.1).

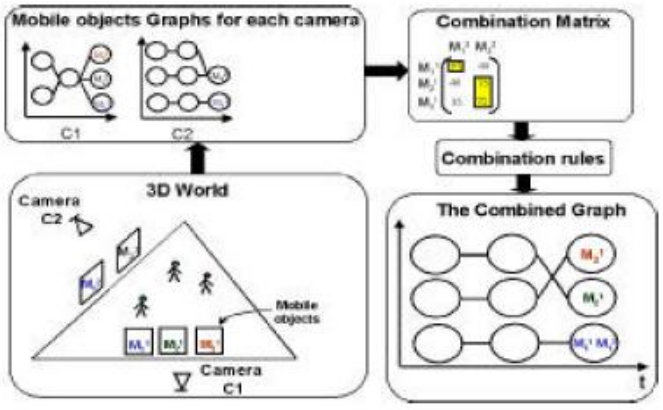


Figure 2.1: All mobile object are combined into a graph.

An actor of a behavior is any scene object involved in the behavior, including static objects (equipment, zones of interest), individuals, groups of people or crowd. The entities needed to recognize behaviors correspond to different types of concepts which are:

The states: a state describes a situation characterizing one or several actors defined at time t (e.g. a group is agitated) or a stable situation defined over a time interval. For the state: "an individual stays close to the ticket vending machine", two actors are involved: an individual and a piece of equipment.

The events: an event is a change of states at two consecutive times (e.g. a group enters a zone of interest).

The scenario: a scenario is a combination of states, events or sub scenarios. Behaviors are specific scenarios (dependent on the application) defined by the users. For example, to monitor train compartments some potential behaviors : "Fighting" "Screaming", "Vandalism" and "Overcrowding".

As described in [13] above described behavior recognition method used a Bayesian Network to model the states, events and scenario methods. And as an alternative instead of a Bayesian Network a AND/OR tree was used. The following was concluded for these 2 methods. Both of these methods need a

learning stage to learn the parameters of the network using ground truth (videos annotated by operators). Bayesian networks are optimal given ground truth but the AND/OR trees are easier to tune and to adapt to new scenes. Another alternative approach to tracking can be done using dynamic Bayesian Networks as described in [35].

A much more sophisticated mass scale system is the PRISMATICA[28], where beside video the audio is fused into the whole system also, this in contrast to the above described recognition model. PRISMATICA is designed to integrate different cameras, contactless smart cards, wireless video/audio transmission, and audio surveillance systems, to monitor different safety and security concerns in railways. The most interesting part about the PRISMATICA system is the MIPSA (modular Integrated Pedestrian Surveillance Architecture). MIPSA is a technical concept that used state of the art technology to support human operators in their task to prevent and detect security-threatening situations. The approached used to fuse different evidences of the different sensors(audio and video) is a Bayesian network. A comprehensive design of this Bayesian network and how it work and fuse information from the different sensors can be found here[28].

The PRISMATICA system show a good concept of applying graphical approach together with Bayesian Networks for fusing the information detected by visual and audio devices(sensors). In most studies about behavior recognition presented in this paper it is clear that Bayesian Networks are a good solution and provides an inference mechanism to fuse the diverse information from different detection devices and provides more descriptive information for the operator to asses incidents.

2.6.2 An Ontology for Situation Awareness

The specific term situation awareness is most commonly used in the community of Human-Computer Interaction. The concerns are to design computer interfaces so that Situation Awareness(SAW) can be achieved automatically in a timely fashion, or that a human operator would be notified in time to act on a given situation. Situation awareness is also used in the data fusion community where it is commonly referred to as situation assessment (Level 2 of JDL model [44]). The term data fusion is used because information originates from multiple sources. So data fusion is the process of combining data to refine state estimates and predictions.

The process of achieving Situation Awareness(SAW) is called situation analysis and the primary basis for SAW knowledge in typically provided by sensors in the area of interest(e.g train platforms).Such a system that assist in situation analysis require the information and ability to know how to represent entities, relations about these entities and their attributes as they propagate in time.

Such a model(or theory) of how the world situation works in the eyes of those doing the analysis. This can be done by an ontology that describes the set of objects and the relationships they have with each other. The relations about these objects must be defined in such a way that any interface using this ontology will be able to represent and capture sufficient information to support high-level reasoning.

The formal definition of SAW as described in [29]. Such a definition is favored because the intention is to be able to formally reason about situations.

Definition: Situation Awareness (SAW) is knowledge of the following:

- A specification of the Goal theory, $\mathbf{T}_g$;
- An ontology, i.e. a theory $\mathbf{T}_o$ of the world;
- A stream of measurements $\mathbf{W}_1$, $\mathbf{W}_2$ for time instances $\mathbf{t}_1$, $\mathbf{t}_2$,;
- At each time instance, the fused theory $\mathbf{T}^t = \nabla_t(T_1^t, T_2^t, ..., T_n^t)$ that combines all the theories that are relevant to the Goal $\mathbf{T}_g$ as well as the fused theory $\mathbf{T}^{t+1} = \nabla(T_1^{t+1}, T_2^{t+1}, ..., T_n^{t+1})$ that combines all the theories that are relevant to the Goal $\mathbf{T}_g$ at some time t + 1 in the future;
- At each time instance t, the fused model $\mathbf{M}^t = \nabla_m(\mathbf{M}_{1.1}^t, \mathbf{M}_{1.2}^t, ..., \mathbf{M}_{2.1}^t, \mathbf{M}_{2.2}^t, ...)$ that combines all models relevant to the Goal $\mathbf{T}_g$ as well as the fused model $\mathcal{M}^{t+1}$ at some time t + 1 in the future;
- Relations $\mathcal{R}^t \subset O^t \times O^t$ relevant at time t, as well as at t+1, $\mathcal{R}^{t+1} \subset O^{t+1} \times O^{t+1}$ among objects (here we consider only binary relations, but the formalization can be extended to include relations of higher arity).

In the definition $\mathbf{T}_o$ (theory of the world) will be defined by the core SAW ontology, which will contain classes to support all the formal symbols in the definition.

The ontology as described in the paper had to satisfy some requirements. First it needed to represent entities and the relationship between them as well as their evolution over time. Another requirement was to be able to express essentially any "reasonable" evolution of entities and relationships. And the last requirement was that it needed to be cheap to build and implement in a working system. Figure 2.2 depicts the main portion of the SAW ontology as a UML diagram. Rectangles are the object classes and the connecting lines are the relationship between these classes. The Situation class (figure 2.2) defines a situation to be a collection of Goals, SituationObjects and Relations. SituationObjects are entities in a situation that can have characteristics (i.e., Attributes) and can participate in relationships. Attributes define values of

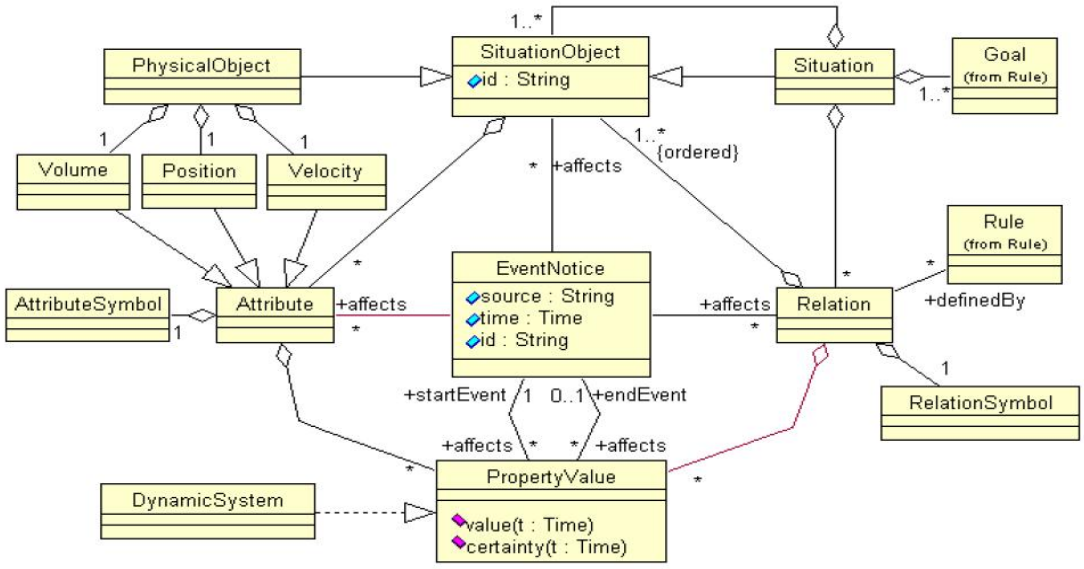


Figure 2.2: SAW Ontology [29].

specific object characteristics, such as weight or color. A PhysicalObject is a special type of SituationObject that necessarily has the attributes of Volume, Position and Velocity. Relations define the relationships between ordered sets of SituationObjects. For example, inRangeOf(X,Y) might be a Relation representing the circumstance when one PhysicalObject, X, is within firing range of a second PhysicalObject, Y. To model time in this model both the Attributes and Relations classes are associated with zero or more PropertyValues which defines two time dependant functions(value and certainty). Value is the actual value and the other is for the certainty assigned. These time values can change when new EventNotices arrive. EventNotices can be seen as new information about the real world from the sensors. This new information can affect the Relation, Attribute or SituationObject. These entities will indicate that something has changed. If more EventNotices arrive over time these entities will change (see figure 2.3).

In figure 2.3 we see at time t1 an *eventnotice-t1* is triggered by some sensor, which affects attribute1 or object1 by assigning its a value and certainty instantiated by *propertyvalue1*. At time t2 a second event generates a *eventnotice-t2*, which assigns new values and certainty in the form of *propertyvalue2* Once propertyvalue2 has been instantiated also marks the end of propertyvalue1. This process repeats as more *eventnotices* arrive at time t.

The*DynamicSystems* (see figure 2.2)class will be implemented as a predictive model. As an example consider the velocity and position attributes of *PhysicalObject*. These attributes are related to each other. The position changes if there is a value for velocity (acceleration, trajectory) and viceversa. So the *Position* at time t+1 depends on the *Velocity* at time t. If no new

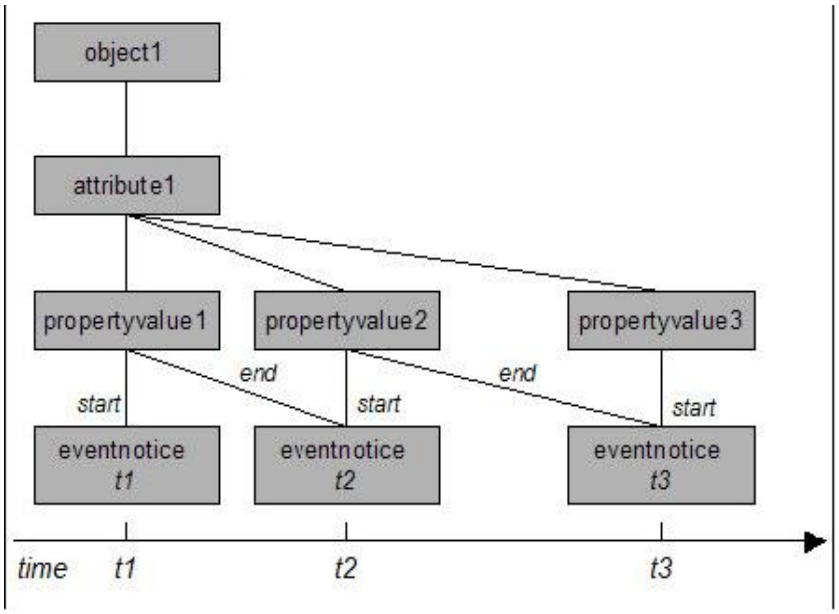


Figure 2.3: EventNotices over time

EventNotice arrives at t+1 it is reasonable to assume the object will still be moving at its last noted speed and direction, all be it with increasing uncertainty as time goes on. Here is where the implemented prediction models in *DynamicSystems* can help to make a projection when no explicit real-world sensor information is available. This core ontology can be extended for specific domains. In [29] there is a overview how this ontology can be extended for battlefield example for the military.

This SAW ontology was designed to be a flexible situation awareness system and also to be extended easily by users. Also SAW would allow end users to use a query language to formulate queries regarding current and possible future situations.

Chapter 3

Introduction to Ontologies

The aim of this chapter is that readers will:

- understand what is meant by the term 'ontology';
- know the range of purposes that an ontology may serve;
- familiar with the main steps in building an ontology and the software tools to support the process of building and using ontologies;
- and know the different ontology languages that are available to represent an ontology;
- know what different types of ontologies can be distinguished.

3.1 What are Ontologies?

The term ontology, originating from the field of Philosophy as detailed in [25], and was adopted by AI researchers to describe formal domain models. Several ontology definitions were provided in the last decades. The most frequently cited definition is that given by Gruber [22] according to who the definition of an ontology was:

> An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an ontology is a systematic account of Existence. For AI systems, what "exists" is that which can be represented.

In order to understand this definition, it must be clear what a conceptualization is. A conceptualization is a structured interpretation of a part of the world that people use to think and communicate about the world. For a biologist such a conceptualization may include that animals can be classified in groups called species and that the animals belonging to a species have similar eating habits.

Based on these eating habits the species can be subcategorized into herbivores, carnivores and omnivores.

In other words, an ontology is a domain model (*conceptualization*) which is explicitly described (*specified*). Later, in 1997 Borst[7] defines an ontology as a "formal specification of a shared conceptualization". This definition requires, in addition to Gruber's definition, that the conceptualization should express a shared view between several parties, a consensus rather than an individual view. Also, this conceptualization should be expressed in a machine readable format (formal). In 1998, Studer [45] merged these two definitions stating that:

> "*An ontology is a formal, explicit specification of a shared conceptualization*".

This definition emphasizes the fact that there must be agreement on the conceptualization that is specified. The reason for including this is that the ability to reuse an ontology will be almost nil when the conceptualization it specifies is not generally accepted.

As consensual domain models, the primary role of ontologies is to enhance communication between humans (e.g., establishing a shared vocabulary, explaining the meaning of the shared terms to reach consensus). As formal models, ontologies represent knowledge in a computer processable format thus enhancing communication between humans, humans and computer programs or two computer programs. Therefore, ontologies are investigated by several research fields in the context of diverse application areas. Nowadays there is an extensive list of references to research fields that have recognized the importance of ontologies, ranging from knowledge engineering to information retrieval and integration. There are also reports on the use of ontologies in several Web related tasks such as Web site organization, navigational support, browsing and searching.

Most ontologies share a few common items such as:

- Concepts, a hierarchical IS-A relation and further relations.
- Some ontologies have constraints, functions or axioms.

An ontology can be as simple as a semantic network, where no distinction is made between concepts and instances, and the only relation possible is of the is-a type, or as complex as CYC [30], which is a large upper-ontology, with a clear distinction between concepts and instances, where multiple inheritance is allowed and where there is an extremely reach set of possible relations. A basic ontology definition could be given by a tuple O:= (C; *is a*; R), where C is a set whose elements are called concepts, is a establishes a partial order on C and R is a set whose elements are called relation names. An example is given in figure 3.1.

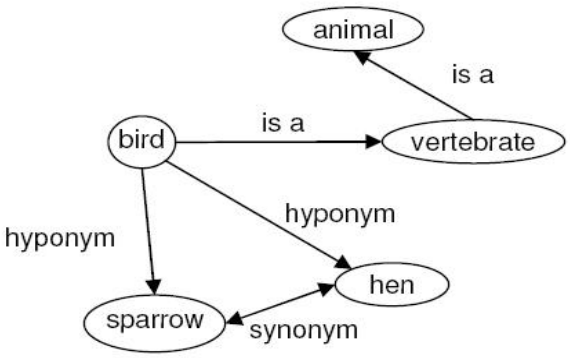


Figure 3.1: An example of a basic ontology.

A graph definition could be given by G = (V,E) where V is the set of vertices and E is the set of edges. An example is given in figure 3.2.

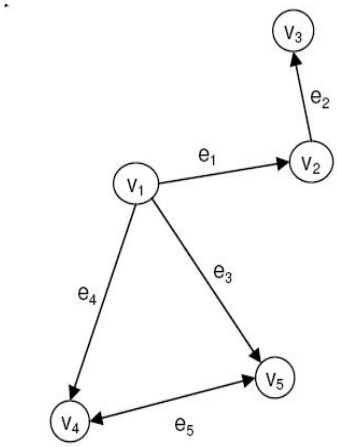


Figure 3.2: A graph structure.

Given the two definitions one can see that graphs fit the basic structure of ontologies very well. Vertices are considered concepts, labeled Edges as relations.

3.2 Uses and roles of Ontologies

From research we believe that five main uses or roles for ontologies can be identified: (1) organize and structure information; (2) reasoning and problem solving; (3) semantic indexing and search; (4) semantics integration and interoperation; and (5) understanding the domain. Below we briefly present each of these roles in general.

3.2.1 Organize and Structure Information

The basic role of ontologies in this case is to organize and structure information in the domain. Ontologies here are tools to describe things or phenomena in the domain of interest. The ontology thus plays the role of vocabulary, answering two main questions: (a) which terms can be used? (i.e., ontology as a lexicon); and (b) which (valid) sentences can be expressed (i.e. ontology as a grammar)? In the field of AI and Law, this role is shown in the use of ontologies to define legal vocabularies. These are typically used to define the terms used in regulations. In this way, the ontologies are not so much legal ontologies but representations of the world or domain the law is working on, e.g. taxes, crime, traffic, immigration, etc.

3.2.2 Reasoning and Problem Solving

The basic role of ontologies in this case is to represent the knowledge of the domain so that an automated reasoner can represent problems and generate solutions for these problems. The ontology here works as the structure of the knowledge base. This use is found in the many expert systems (problem solvers) and decision making systems developed in AI. In using ontologies for this role, secondary goals are to create knowledge bases that are reusable, efficient, explainable, modular, etc. Indeed, one can argue that the use of ontologies in AI comes from research in the late eighties and nineties that aimed at improving knowledge engineering by attacking these roles by creating "well-structured" knowledge bases that will not only solve the problem at hand but be more maintainable, easier to extend, etc. In this sense, ontologies in this use are very much an engineering tool. This role of ontologies implies the use of an inference engine that is used to conclude specific goals.

3.2.3 Semantic Indexing and Search

The basic role of ontologies in this case is to represent the contents of documents or other "soft" knowledge sources (picture, movies, etc.). The ontology here works as a semantic index of information, that enables semantic search for content. There are many organizations that produce vast amounts of knowledge in the forms of documents, charts, schemas, etc. There is a key need to organize and be able to find these documents. Ontologies can be used to represent and search semantically the content of documents - to go beyond word or keywords.

3.2.4 Semantic Integration/Interoperation

The basic role of ontologies in this case is to support applications to exchange information electronically. The ontology here works as an *interlingua* that defines a (narrow) vocabulary to be used to interchange information.

3.2.5 Understand the Domain

The basic role of ontologies in this case is to provide a view of what a domain is about–to try to make sense of the variety of knowledge in that domain. The ontology here works as a map that specifies what kinds of knowledge can be identified in the domain. This type of ontology can be used as a basis for designing specialized representations. Because it tries to get close to the nature of the domain, it frequently connects and draws from theories of that domain. These types of ontologies have been called core ontologies. Some of these core ontologies are also used or at least designed for supporting reasoning and problem solving.

3.3 Ontology Languages

Many languages have been developed over the years in order to promote knowledge sharing and data integration. However, we will only briefly discuss three such language models here specifically developed for the development of ontologies. All these ontology languages are based on Resource Description Framework(RDF) triplets and support reasoning capabilities that are both key aspects of the recommendations set forth by the Semantic Web[11]. The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.

RDF's model for representing data about resources is that of Object-Attribute-Value(O-A-V) triplets(see Figure 3.3. A resource description in RDF is a list of statements(triplets, each expressed in terms of a object, one of its properties (attributes), and the value of the property. The value can be a literal(text), or another resource. This is an important concept because all the current ontology languages are based on this triplet principle.

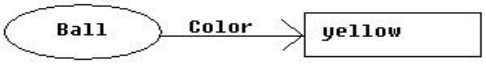


Figure 3.3: An example of O-A-V triplet.

The ontology languages in question include the previous W3C [11] recommendation, DAML+OIL, the older OKBC ontology language and the current ontology language of choice, Ontology Web Language, OWL [42].

3.3.1 DAML+OIL

Darpa Agent Markup Language (DAML)[14] is an ontology language that was developed by the RDF Core Working Group in order to represent ontological representations more explicitly than XML, RDF, and RDF Schema. DAML+OIL is the extension of DAML, which was later developed. DAML+OIL, the previous W3C standard in ontology language combines DAML and the Ontology Inference Layer (OIL). DAML+OIL consists of class elements, property elements, and instances. DAML+OIL can use an imports statement to reference another DAML+OIL ontology. DAML+OIL also divides the domain into datatypes and objects.

This ontology language supported the field at the time it was recommending, but could not keep up with the growing need for more expressive ontologies because of the limited restriction and concept support. Thus, OWL took the place of DAML+OIL as the semantic web standard.

3.3.2 OKBC Ontology Language

The Open Knowledge Base Connectivity is a protocol for accessing knowledge bases. The OKBC knowledge model is the implicit representation formalism that underlies all the operations provided by OKBC. It supports an object-oriented representation of knowledge. The knowledge model is extensive and well defined, and has several implementations. In this section we give an informal description of the OKBC knowledge model.

As OKBC is a frame-based representation mechanism, a frame is the central object in the model. A *frame* represents an entity in the domain of discourse. There are three main types of frames: *class frames*, which represent sets of entities, *slot frames*, which represent binary relations, and *individual frames*, which represent single entities. If entities are member of a class, they are said to be *instance_of* that class. The other way around, the class is called the *type_of* that instance. A class can be a *subclass* of another class: if this is the case, then all instances of the subclass are also instances of the other class. All frames can be related via slots to other frames or constants. Slots that are associated with a frame are called own slots of such frame. For example, an individual frame "George Bush" can be related with the own slot "president_of" to the frame "United States". Class frames can also have *own slots*, although this is much less common.

Besides own slots, class frames can also be associated with a collection of *template slots*, that describe slots that are considered to hold for all members of that class. For example, the class "blue_ball_point_pens" can have the template slot "ink_color" to the constant "blue", meaning that every *instance* of "blue_ball_point_pens" should have the value "blue" for the slot "ink_color".

Template slots of a class inherit to its subclasses. Slots that are related to a frame can have associated with them a set of *facets* and *facet* values. Facets and facet values describe characteristics of the combination of a frame and a slot. For example, the facet "value-type" and value "President" associated with the slot "rules" in the class frame "Republic" specifies that that value of the slot "rules" for each instance of a republic should be an instance of a president (as opposed to monarchies, where the slot"rules" should relate to an instance of a king or queen).

The OKBC knowledge model contains a number of standard facets, concerning the *value-type*, the *inverse* relation, the *cardinality* and the *equivalence* of the slot among others.

Figure 3.4 shows a UML class diagram of the main elements of the OKBC knowledge model. For the sake of clarity, we made a few simplifications. First, we modeled constraints as a separate class, although constraints are either specified by a facet or are defined as global constraints on a slot. In the OKBC model, all constraints exist in two different variants: both as a facet and as an slot on a slot. As a second simplification, we didn't show all types of constraints. The constraints that are missing in the picture are:

- disjointness of slots;
- numeric minimum and numeric maximum for slot values;
- subset of values of a slot;
- collection type of multiple slot values: multiple values are either treated as *set, list or bag.*

3.3.3 Ontology Web Language, OWL

The Ontology Web Language (OWL)[42] is a successor to DAML+OIL (figure 3.5) and is the current W3C standard for ontology languages and has been extended to provide more explicit description logics. OWL also provides three increasing levels of expressivity in OWL Lite, OWL DL, and OWL Full respectively. This allows users to define their own needs for expressivity and chose a language version that best supports their needs. A short description of the 3 sub-languages of OWL:

- **OWL Lite** supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. Owl Lite also has a lower formal complexity than OWL DL.

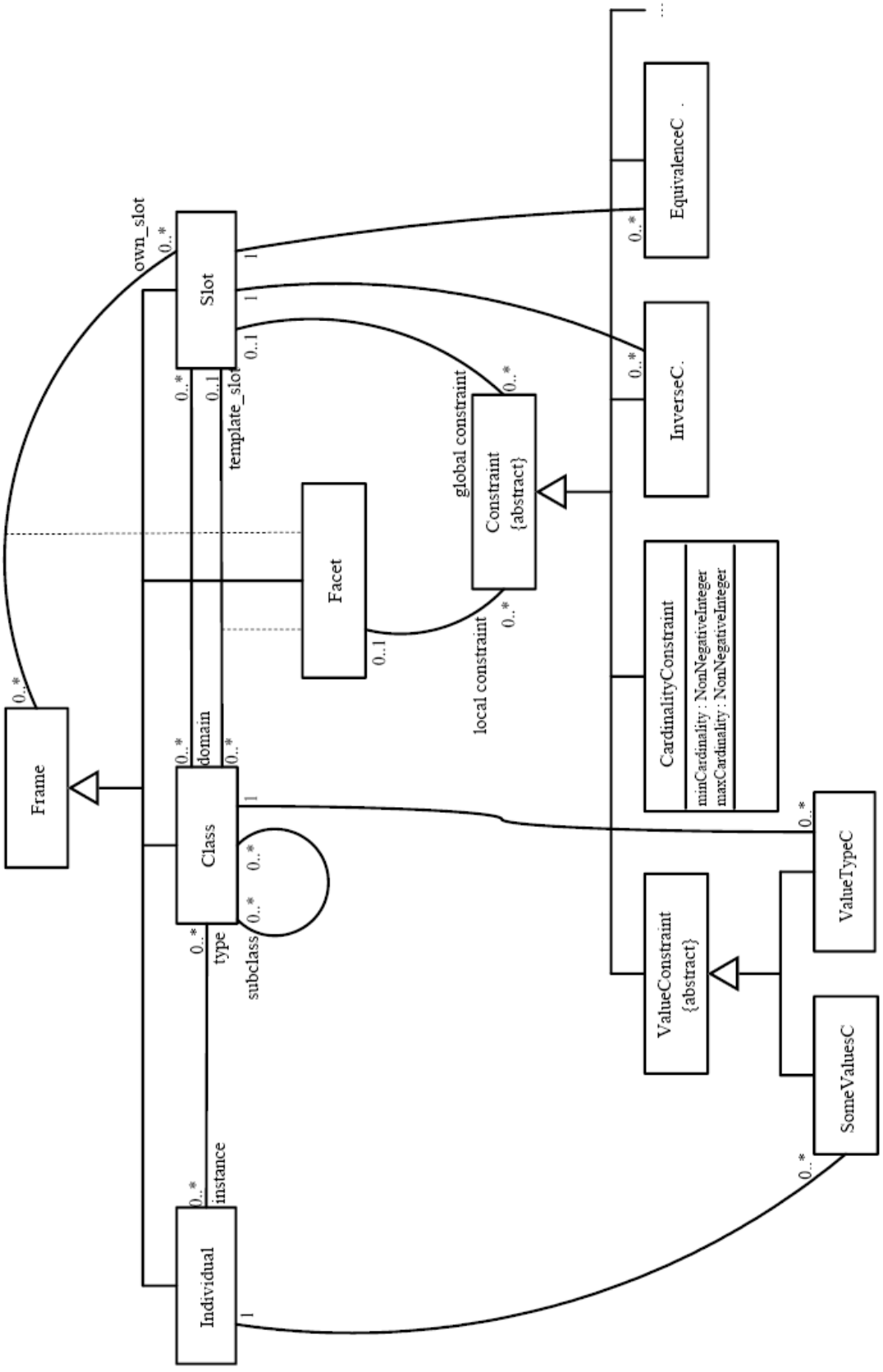


Figure 3.4: A simplified representation of the OKBC knowledge model.

- **OWL DL** supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with description logics, a field of research that has studied the logics that form the formal foundation of OWL.

- **OWL Full** is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full. This is the reason that OWL Full is seldom used and OWL DL is mostly the chosen one. More concretely, the DL variant of OWL (OWL-DL) is chosen because, although it is constrained in order to be managed by Description Logic (DL) reasoners, such reasoners guarantee that ontologies can be put reasoned over in an efficient way. Moreover, existing tools can be used to make the implementation quite straightforward.

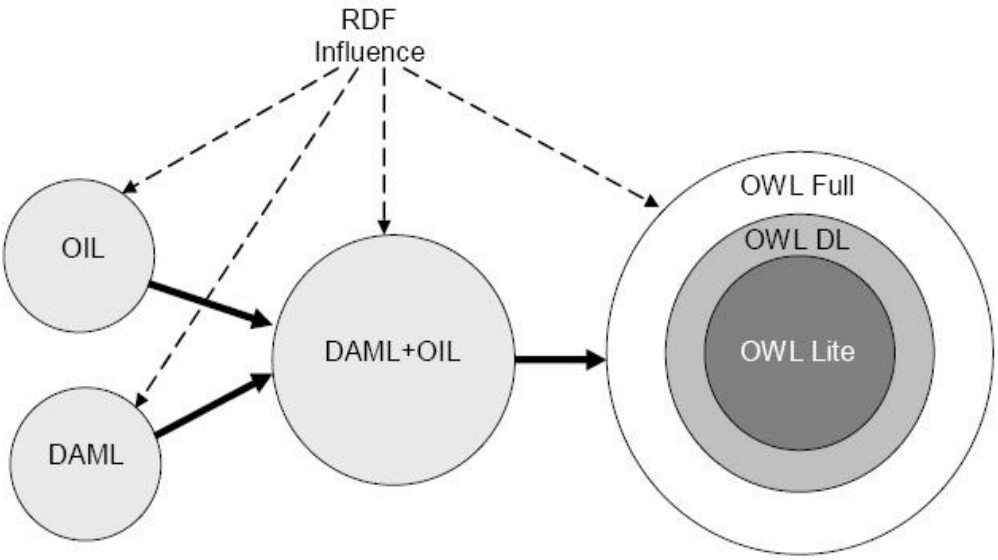


Figure 3.5: The creation of OWL.

The OWL syntax employs Uniform Resource Identifiers(URIs)[24] for naming and implements the description framework for the Web provided by RDF to add the following capabilities to ontologies: the ability to be distributed across

many systems, scalability to Web needs, compatibility with Web standards for accessibility and internationalization, and openness and extensibility.

3.3.3.1 Changes compared to DAML+OIL

Changes from DAML+OIL to OWL include various updates to RDF and RDF Schema from the RDF Core Working group, DAML+OIL restrictions were removed, and various properties and classes were renamed in OWL syntax. In addition, other properties like for example *Owl:SymmetricProperty* were added and DAML+OIL synonyms for RDF and RDF Schema classes and properties were removed, as well as added properties and classes to support versioning and unique names assumptions. The Ontology Web Language employs the most recent version of RDF Semantics, which thus replaces some semantic terms identified in DAML+OIL. RDF and RDF Schema updates include: allowing cyclic subclasses, handling multiple domain and range properties as intersections, changing namespaces, and implementing XML Schema datatypes and new syntax for list functions. Figure 3.6 shows a example excerpt of a statement and how it would be written in OWL RDF/XML syntax. Overall, the changes and updates that have been implemented from DAML+OIL to OWL have made the Web Ontology Language a more expressive ontology language standard.

```
<owl:Class rdf:ID="Student">
    <owl:intersectionOf rdf:parsetype="Collection">
        <owl:Class rdfs:about="Person" />
        <owl:Restriction>
            <owl:onProperty rdf:resource="enrolledIn" />
            <owl:minCardinality rdfs:datatype="&xsd;Integer"> 1
            </owl:minCardinality>
        </owl:Restriction>
    <owl:intersectionOf>
</owl:Class>
```

Figure 3.6: OWL Excerpt: a Student is a Person who is enrolledIn at least one thing.

3.3.3.2 Changes compared to OKBC

OWL differs from OKBC in a number of aspects. Besides a few terminological differences (the most notable is that a slot is called a property), we can see the following differences.

- In OWL, the set of slot constraints is divided into global and local constraints. That is, a specific constraints is either a global constraint that holds for all values of the slot (e.g. "functional"), or it is a local constraint that only restricts the values of a slot when used in a specific

class (e.g. "has-value"). In OKBC, all constraints can be applied both globally and locally. The following constraints can be applied locally in OKBC, but only globally in OWL:

- inverse of a slot;
- equivalence of a slot;
- values are subset of the values of another slot (called subslot if applied at global level).

- In OWL "slotfacetvalue" triplets are classes themselves. That is, a constraint on a slot, called a property restriction, defines a class. For example, the property restriction "agehasvalue27" defines the class of things that have the value "27" for the slot "age", i.e. the class of all 27 years old things.
- Slots in OWL are divided into slots that can have instances as their value, and slots that can have data type values.
- Classes and individuals in OWL can be declared as equivalent or disjoint.

Figure 3.7 shows a UML class diagram of the OWL meta model. Based on these differences, we can not conclude that one knowledge model is contained in the other model. However, when we look carefully at the differences, we can see that the elements of OKBC that are missing in OWL are quite rare. For example, it is difficult to think of examples or a practical usage of the local equivalence constraints on slots, or a local inverse constraint. It is likely that these constructs are present in OKBC for reasons of symmetry with the global constraints. The disjointness of slots seems to be the most useful construct that is missing in OWL. Besides these aspects, we can consider the OWL knowledge model as almost a superset of OKBC for practical applications.

3.4 Ontology Tools

3.4.1 Protégé Ontology Editor

Protg is a methodology for building knowledge-based systems from three classes of reusable components:

- domain ontologies, or models of the concepts in an application area and relations between those concepts;
- associated knowledge bases containing domain facts; and
- problem-solving methods, or algorithms that apply generic reasoning patterns to domain knowledge.

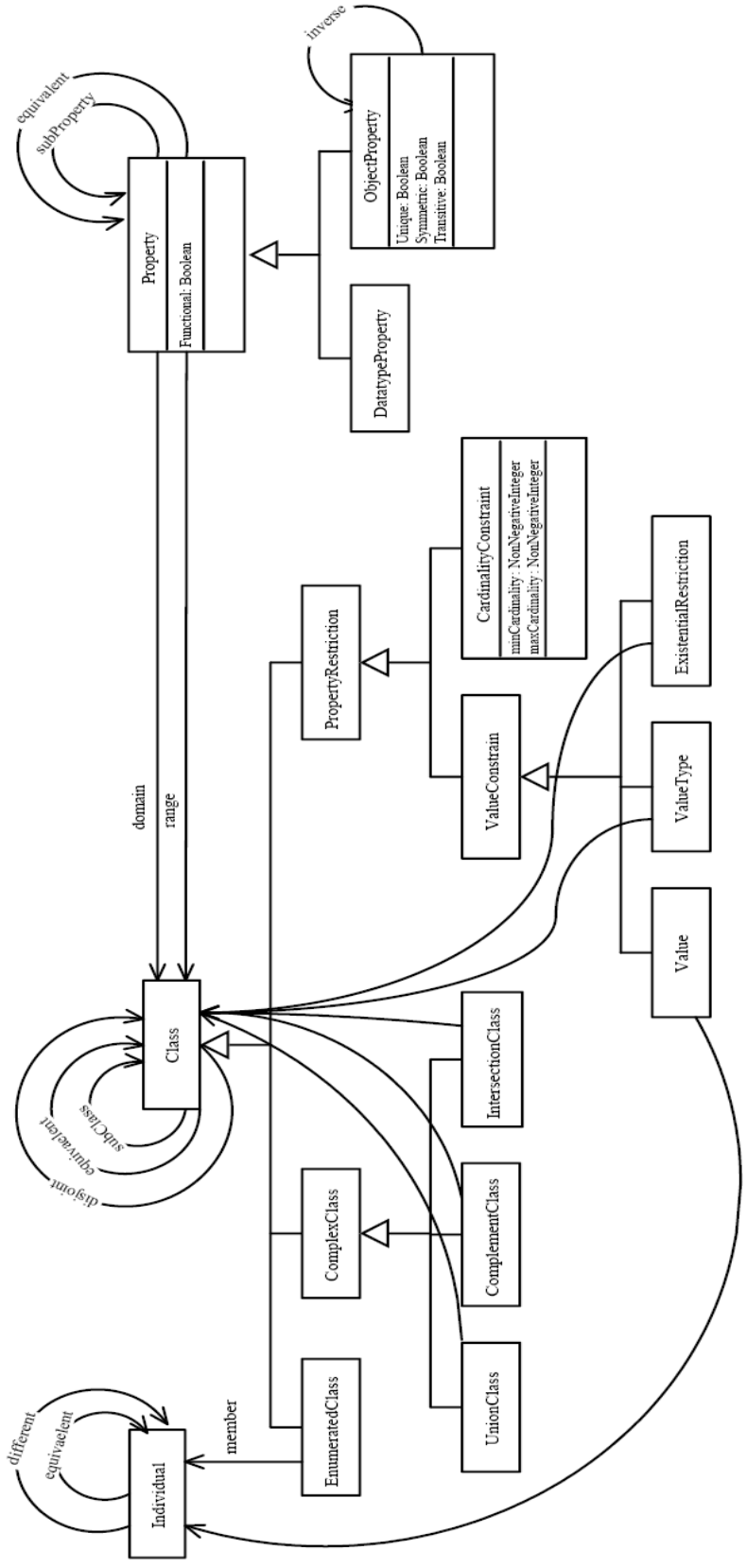


Figure 3.7: A UML representation of the OWL meta model.

Protégé [17] is a free, open source ontology development framework that gives a growing user community a tool suite to construct domain models and knowledge- based applications. Protégé implements a knowledge model compatible with the Open Knowledge Base Connectivity protocol, designed for interoperability among frame-based systems. In a frame-based modeling representation, an ontology consists of a set of classes organized in a subsumption hierarchy to represent a domain's salient concepts, the *properties*-or slots-associated to each concept, and a set of *instances* of those classes-individual exemplars of the concepts that hold specific values for their properties. Protégé also supports other formalisms for representing knowledge bases, such as the Semantic Web languages RDF and OWL. Protégé provides both a wide set of user interface elements for knowledge modeling and entry and the capability to include custom-designed plug-in elements as application extensions. Protégé is not only a environment for building ontologies, it's also a server that can provide knowledge encoded in ontologies to any piece of software invoking it.

Along with the above and that Protégé is the most used ontology editor, we chose the Protégé ontology editor and acquisition system [17] for the design of our ontology. Protégé provides an intuitive interface for developing ontologies by supporting multiple design panes for hierarchical design, property design, restriction construction, comment and definition development, and disjoint function construction. Protégé supports a number of ontology languages, including OWL. The Protégé OWL plugin allows for a supported development of OWL ontologies through its use of the rules and syntax of the OWL language as well as support for reasoning. The ontology interface, depicted in Figure 3.8, includes OWL Classes, Properties, Forms, Individuals, and Metadata tabs. The OWL Classes tab shown in Figure 3.8 provides the basic ontology development interface. This interface includes an Asserted Hierarchy toolbox for creating hierarchies, a Comment box to include additional descriptions of entities, Asserted Conditions hierarchy which displays the restrictions of each class, Annotations which include additional annotation development, Properties which display the properties that are defined in the Properties tab, and Disjoints toolbox which aids in defining classes as disjoint. This robust and intuitive interface provides an outstanding tool for creation of ontologies while the backend ontology language rule and syntax control mechanisms allow for easy development and checking of not only the design of an ontology, but also the syntax necessary for the ontology to communicate its knowledge with other systems.

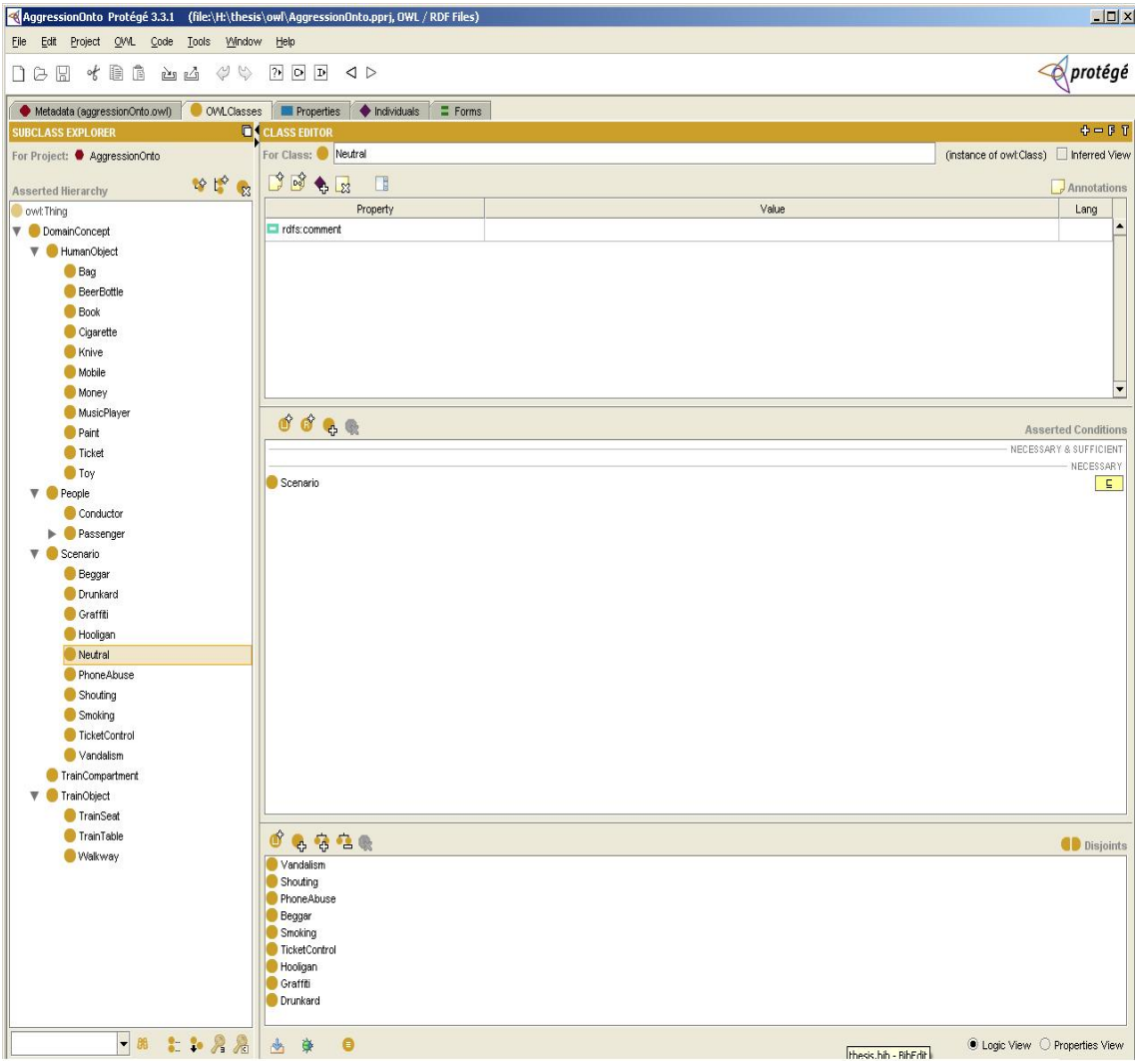


Figure 3.8: The Protégé OWL User Interface.

3.4.2 RacerPro Reasoner

A reasoner is important in ontology development due to its ability to infer logic from existing entities with consistency checking and classification for subsumptions. RacerPro[39] is a commercial product, but fortunately there is a educational license which allows it to be used with all the features for universities and research purposes. The RacerPro reasoner supports the OWL ontology language and can also be easily integrated with Protégé and thus was a good solution for a reasoner. This reasoner supports Abox and Tbox reasoning over classes(concepts) and individuals(instances) respectively. In our case, Tbox reasoning is an important feature since the proposed ontology contains high level concepts, or classes, to describe the domain. RacerPro is able to provide the high level reasoning capabilities by testing for concept satisfiability and class consistency. Protégé and RacerPro provide a sound basis for design and inferring ontologies. However, each tool was tested while in

development; therefore, occasional bugs in the systems would cause the need for additional tools to check syntax and validity of the ontology.

3.4.3 WonderWeb OWL Ontology Validator

The WonderWeb OWL Ontology Validator [46] was the tool of choice to check the syntax and validity of the ontology developed here. This validator was developed by Sean Bechhofer of the University of Manchester and Raphael Volz of the University of Karlsruhe as part of the EU IST Project WonderWeb [37]. The WonderWeb OWL Ontology Validator was created in an effort to provide classification of OWL ontologies into OWL Lite, OWL DL, or OWL Full. Not only was the validator utilized for those purposes of classification, but the detailed responses to the validation were also utilized as a method to analyze and recover from errors in the ontology syntax. This was a valuable addition to the tool set already available because in many cases, when errors occurred within Protégé, then they could be resolved with the help from the validator. The detailed response of the WonderWeb validator was used to distinguish the cause of and correct the errors.

3.5 Summary

In this chapter we give a brief overview of what ontologies are and their use in the Artificial Intelligence field. Some important ontology language models were described of which OWL is the W3C standard at the moment. It is important to note that OWL also supports the construction of distributed ontologies, which is beneficial in many ways. The Semantic Web initiative has invoked the creation and sharing of many ontologies which are distributed across the web. When creating an ontology for a given use, it is most efficient and effective to rely on the expertise of others and previous models in order to provide a more robust representation of a domain. Thus, the integration of distributed ontologies becomes an important design implication. Also, as the breadth and depth of the individual ontology increases, the ability to manage the information contained within the knowledge base also increases. Thus, the support of a distributed ontology system where specialized ontologies can be maintained as separate entities becomes an attractive option. One advantage of a distributed ontology is that it can be collaboratively created and easily maintained over time. Specialists in their field of expertise can gain access to a particular part of the ontology in order to update and revise it as they see appropriate without interrupting the integrity of the top-level system ontology. The ability to collaborate with many different professionals adds to the depth and breadth of any ontology and will result in better reasoning and query capabilities.

Not only does OWL provide better expressivity and support for distributed

ontology systems, but stable programs have also been developed to provide editing, reasoning, and inferencing capabilities for the Ontology Web Language. The above discussion clearly outlines the expressivity and support of OWL compared to DAML+OIL. The new World Wide Web Consortium standard is clearly the choice for the aggression ontology proposed in this thesis.

Chapter 4

The design of the aggression ontology

In this chapter we will analyze the train domain and with that information design the aggression ontology based on the Ontology Web Language (OWL). The purpose of the ontology is two fold:

- Ontologies capture potential objects and potential relations; that is to say, they do not describe what is in the world but rather what can be in the world, they can be used to annotate markup descriptions of instances of the world. In other words we can meaningfully annotate our data using object and relations specified in the ontology. We can also use the data to verify the completeness of the ontology.
- Ontologies provide a means for describing and reasoning about sensor data, objects, relations and general domain theories. For fusion systems that combine data from very dissimilar sources or perform fusion tasks that are more diverse, ontologies can be more effective then ad hoc techniques.

4.1 The domain and scope

To start designing the aggression ontology, it has to be pointed out that there is no one "correct" way or methodology for developing ontologies. For our aggression ontology we adopted an iterative development approach: we started with a rough first pass at the ontology. We then revised and refined the evolving ontology and fill in the details.

To determine the scope and domain of our ontology there are several basic questions to answer:

- What is the domain that the ontology will cover?

- For what we are going to use the ontology?
- For what types of questions the information in the ontology should provide answers?

Although the answers to these questions may change during the ontology-design process, they can help at any given time to limit the scope of the model and thus keep the design within some boundaries. The domain that the aggression ontology will cover is of course the train compartment. The ontology we are designing will be used to assist in transforming it into a Bayesian network to help us reason about the observed data in the train environment, which will determine the scenario of the train at a given time.

In a train environment there are several types of objects that can initiate aggressive behavior. We will discuss these types of concepts in the following sections along with the different aggressive scenarios that can occur in a train compartment as they will be the main concepts for the ontology. Also the relations between the concepts will be defined properly.

4.2 Ontology class hierarchy

The important task now is to define our classes for the ontology. Figure 4.3 shows the top-level class hierarchy of the ontology. These six top-level classes are a good start to define the scope of this domain. If you look at the classes the logic is simple. The *DomainConcept* class contains six subclasses: *TrainCompartment*, *TrainObject*, *Person*, *HumanObject*, *Situation* and *Scenario*. These describe all the concepts we can have in our train environment to analyze the situation later when transforming it to a Bayesian network. All these classes will have subclasses which are a specialization of that class and relations between them. Each subclass will be discussed later in detail.

But lets discuss the 6 central classes in our ontology first. The **TrainCompartment** class encompass all of the objects that are in a train compartment. Figure 4.1 and 4.2 show an example of a train compartment. The assumption is that once, aggression detection in a single compartment is possible, it is possible to extend it to an entire train, which consists of multiple compartments. We could make a sub-ontology for the complete train compartment consisting of the sections, doors, windows, toilets etc., but we want to keep our focus to the aggression part. In the future the **TrainComparment** class could be a sub-ontology describing all the other aspects.

We modelled the **TrainObjects** class which we limited to some static objects in the train compartment. For example seats and tables. These **TrainObjects** are inclused because they could be the victim of damage due to paint, fire, etc. **HumanObject** class are all the physical objects which can trigger events and

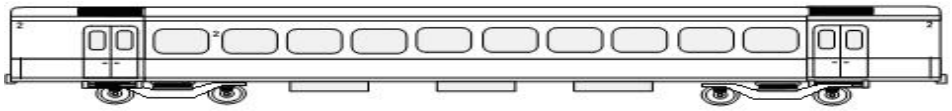


Figure 4.1: Side view of a BeneLux train compartment.

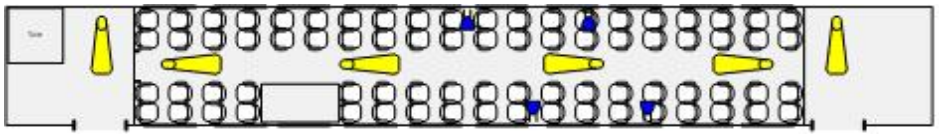


Figure 4.2: Top view of the interior of a BeNeLux train compartment.

cause a situation in the train. These objects can be static objects or dynamic objects. In the next section we model some examples of the human objects.

An important entity in the dynamic context is the actor. The actor is a human who is capable of manipulating (generating events in) the dynamic context (by actions and behaviors). The actor is modelled by the **Person** class. Aggressive behavior involves a human aggressor and one or more human or non-human victims. A situation that occurs by a combination of an aggressor and possibly one or more humanObjects will be a scenario. According to our ontology, a scenario consists of a sequence of situations. An action or event caused by a actor and possibly using humanObjects could result in another situation being triggered and possible switching the scenario to maybe a even more dangerous one.

For our ontology we used the combination approach to make our class hierarchy, because we have a semantic view of the more generalized concepts(e.g. train objects, human objects) as well as the most specific concepts(e.g. neutral, annoyances, danger, damage etc.). So we need to work towards the middle concepts which tends to be the more descriptive concepts in the domain.

Figure 4.3: Top-level class hierarchy.

Let us take a look at the subclasses of the top-level classes. Figure 4.5

illustrates the same complete aggression ontology in a class diagram, so we can see how it is structured. If we look closely at each of the subclasses we can clearly see how each top-level class is specialized. Every *Scenario* in a train will contain a *Situation* or situations in the train compartment. It's easy to see that there will be a relationship between these *Situation* and *Scenario* classes. We have limited the *Situation* class to a couple of the frequently occurring ones as given by the Dutch Railways. These can be easily extended in the future with more situations. The *Scenario* subclass is in fact the end(leaves) of the hierarchy when seen in a top-down figure. Scenarios can be Annoyances, Damages, Danger, Neutral and Sickness. This means that one or more situations can be categorized into one of these scenarios. Table 4.1 illustrates which situations belong to its corresponding scenario category. Below we give a finer definition of the **Scenario** classes:

- Annoyances: This is a combination of irritation perceived by other passengers. For example loud shouting, talking too loud on a mobile phone etc.
- Danger: This indicates that a serious situation occurred. For example fighting, terrorist attack etc.
- Damages: This indicates for example the presence of hooligans damaging the train property.
- Sickness: This scenario indicates that an actor in the train compartment is not feeling well. For example vomiting or fainting.
- Neutral: This is just when everything is quiet in the train compartment and no strange event is taking place. This is mostly the scenario that railways want in their trains.

The classes alone will not provide enough information to answer how they are related to each other. Once we have defined the classes, we must describe the internal structure of the concepts. These object properties or slots as they are called in ontologies will be discussed in the next section.

Table 4.1: Situation-Scenario combination.

Situation	Scenario
Abandoned_Lugguage	Danger
Beggar	Annoyance
Drunkard	Annoyance
Fainting	Sickness
Fighting	Danger
Graffiti	Damage
Phone_Abuse	Annoyance
Screaming	Danger
Smoking	Annoyance
Theft	Annoyance
Vandalism	Damage
Vomiting	Sickness

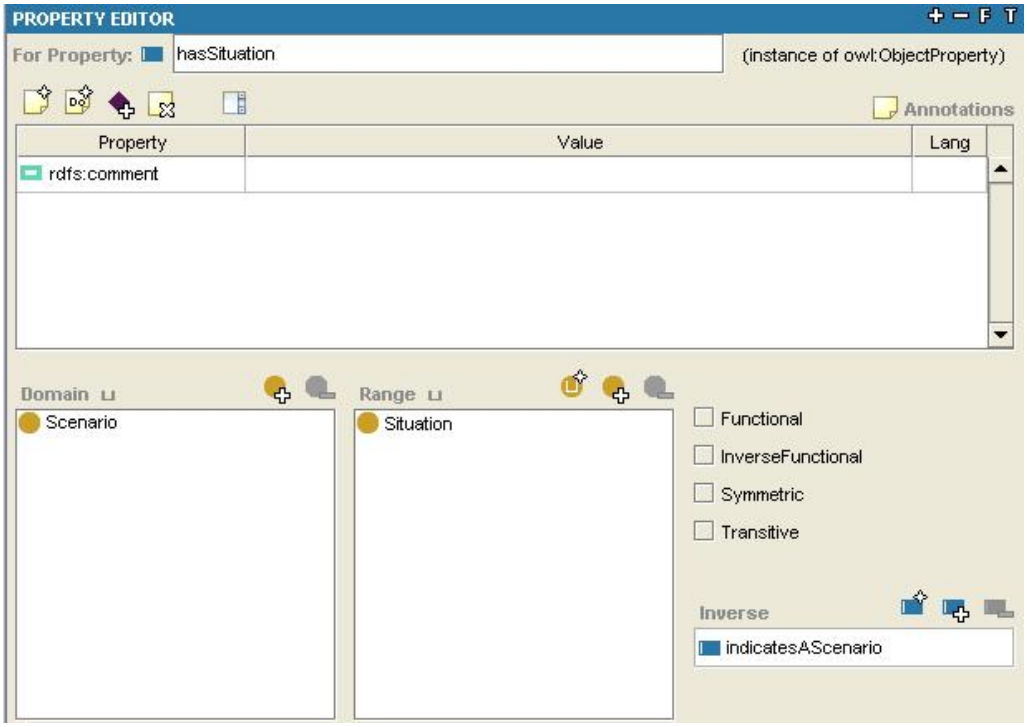


Figure 4.4: Definition of hasSituation slot in Protégé.

Figure 4.5: The aggression ontology class diagram.

```
<owl:ObjectProperty rdf:ID="hasObject">
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <owl:Class rdf:about="#Person"/>
                    <owl:Class rdf:about="#Situation"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
        <rdfs:range rdf:resource="#HumanObject"/>
        <owl:inverseOf rdf:resource="#isObjectOf"/>
</owl:ObjectProperty>
```

Figure 4.6: hasObject in OWL code

4.3 Properties of classes

Slots or properties can have different facets describing the value type, allowed values, the number of values (cardinality), and other features of the values the slot can take. Relations between classes are modelled as object properties. The only exception is the formal relation ”Is-a” which is modelled with the ”owl:subclassOf” OWL relation. As in the case of semantic types (nodes that constitute the ontology), a taxonomy has been built for relations.

Domain and range of a slot

Slots(also known as properties) may have a *domain* and a *range* specified. Properties link individuals(instances) from the *domain* to individuals from the *range*. Allowed classes for slots of type Instance are often called a **range** of a slot. In the example in Figure 4.4 the class *Situation* is the range of the *hasSituation* slot. Some systems allow restricting the range of a slot when the slot is attached to a particular class. The classes to which a slot is attached or a class which property a slot describes, are called the **domain** of the slot. The *Scenario* class is the domain of the *hasSituation* slot (Figure 4.4). In systems where we attach slots to classes, the classes to which the slot is attached usually constitute the domain of that slot. There is no need to specify the domain separately. Table 4.2 show all the object properties that are defined in our ontology. Illustrated is also the domain and range classes which are related by these properties. In Figure 4.8 shows the complete diagram with each object property and their corresponding domain and range classes.

Figure 4.6 illustrates how the objectProperty hasObject is defined in OWL code. You can clearly see the domain and range classes(if any) and if there is an inverse objectProperty(slot) defined for it.

Inverse Slots

A value of a slot may depend on a value of another slot. For example, if a *Scenario* hasSituation *Situation*, then a *Situation* indicatesAScenario from the

Scenario class. These two relations, hasSituation and indicatesAScenario, are called **inverse relations**. Storing the information "in both directions" is redundant. When we know that a scenario has a situation, an application using the knowledge base can always infer the value for the inverse relation that the situation *indicatesAScenario*. However, from the knowledge-acquisition perspective it is convenient to have both pieces of information explicitly available. This approach allows users to fill in the situation in one case and the scenario in another. The knowledge-acquisition system could then automatically fill in the value for the inverse relation insuring consistency of the knowledge base(see also Figure 4.7).

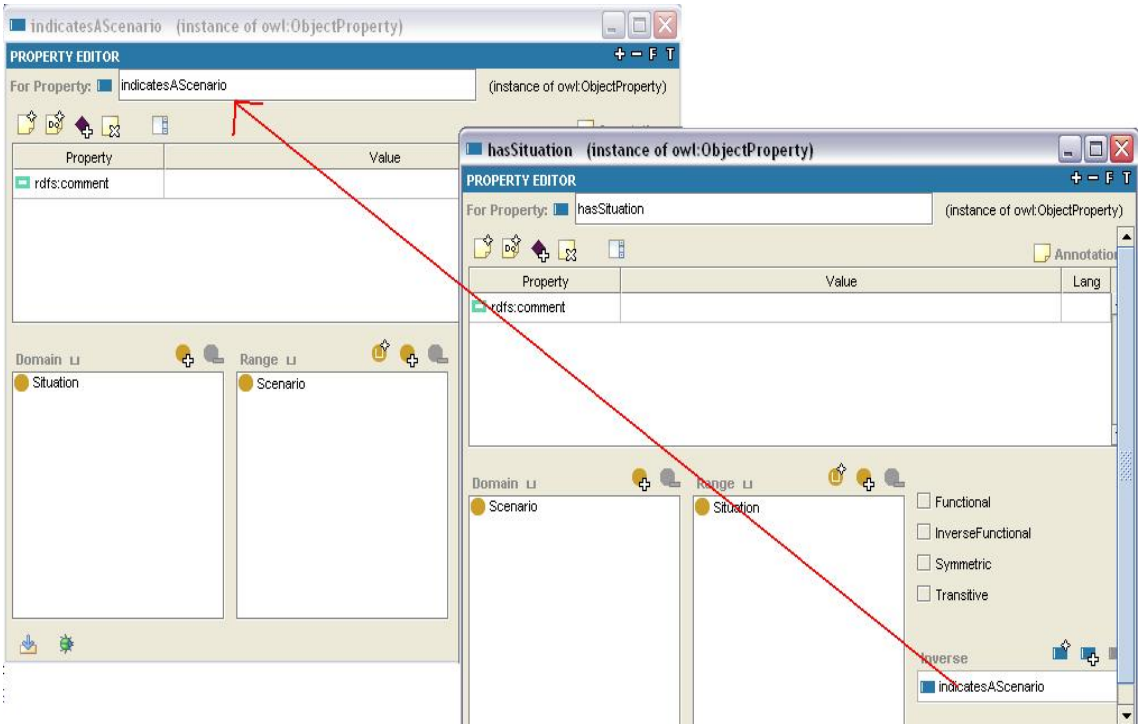


Figure 4.7: Inverse relations (hasSituation-IndicatesAScenario).

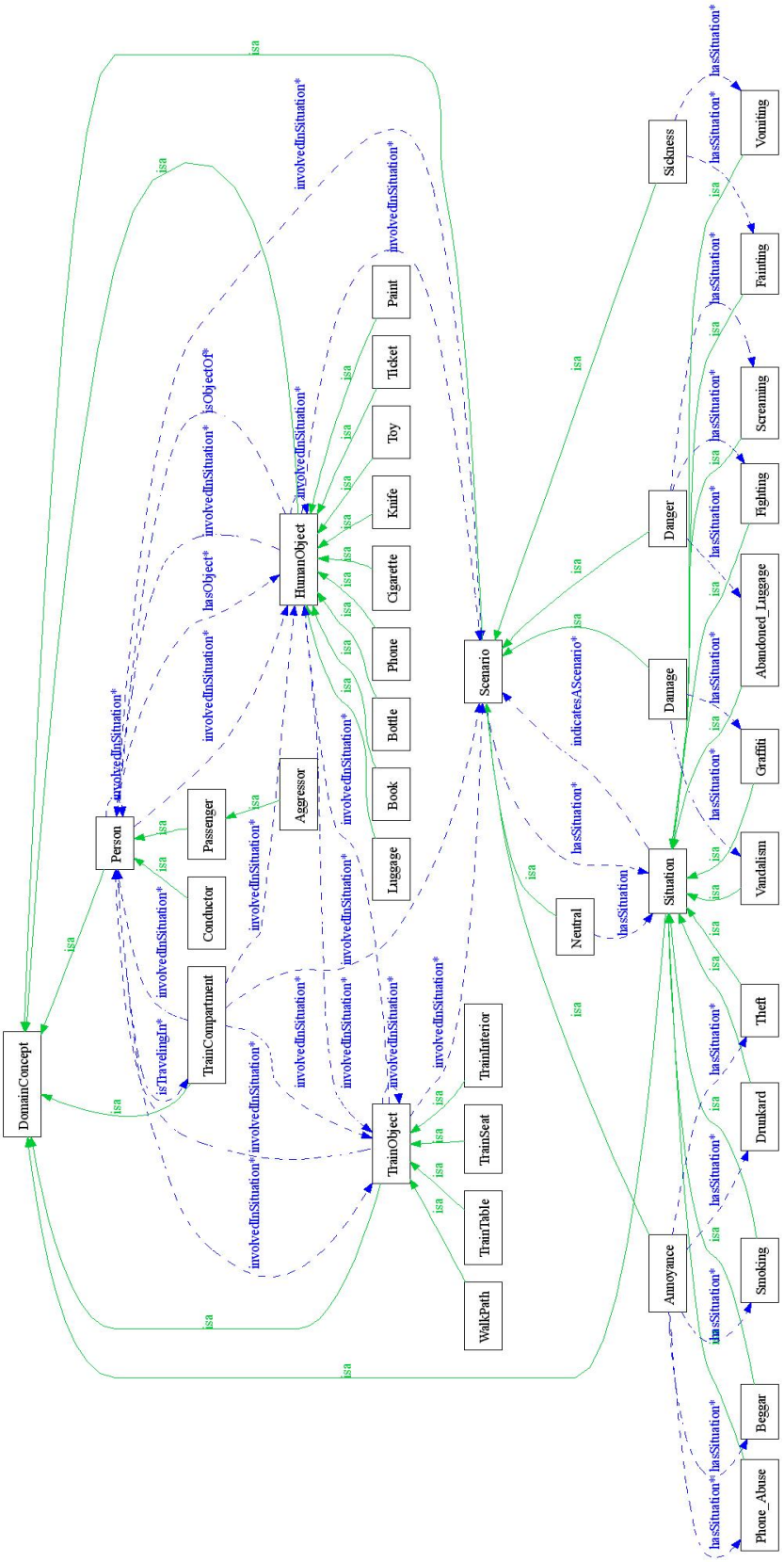


Figure 4.8: A complete view of all the object properties in the Ontology.

Table 4.2: All Object Properties of the Aggression Ontology.

Object Property	Domain	Range
hasSituation	Scenario	Situation
indicatesAScenario	Situation	Scenario
hasObject	Person , Situation	humanObject
isTravelingIn	Person	TrainCompartment
isObjectOf	humanObject	Situation, Person
involvedInSituation	Person , TrainObject, humanObject	TrainObject, humanObject, Scenario , Person

Table 4.2 show the main objectproperties we defined in our ontology. Let us give a definition of these properties so the semantics are clear.

- **hasSituation:**this property relates a Scenario to a Situation. For example we have the Sickness scenario and we can relate this via the hasSituation property to Vomiting situation. Here Scenario is the domain and Situation is the range.
- **indicatesAScenario:** this property is the inverse property of hasSituation. Here a situation is related to a specific scenario.
- **hasObject:** this property relates individuals(instances) of the Person and/or Situation class to humanObject. For example a person can possess instances of a humanObject and/or a occurring Situation may involve instances of a humanObject.
- **isObjectOf:** this property is the inverse property of the hasObject class.
- **involvedInSituation:** this property can relate individuals of the domain classes Person, TrainObject and humanObject to individuals of the range classes of this property which are trainObject, humanObject , Scenario and Person.
- **isTravellingIn**: this property relates individuals of the Person class to the trainCompartment. It means that persons are traveling in a traincompartment. So they are contained in the train compartment.

4.4 Defining the classes by using restrictions

Having created some object properties we can now use these properties to describe and define our aggression ontology classes. In OWL, properties are used to create *restrictions*. In Appendix C we give some more background information of the different restriction types in OWL. In this section we discuss how we implement restriction in our aggression ontology.

4.4.1 Implementing restrictions

In our ontology we have three classes(and their subclasses) which we need to define using restrictions. These are the *Person*, *Situation* and *Scenario* classes. In our modeling software *Protégé* restrictions can be described in a easy fashion to our ontology using the "Conditions Widget". Let start with the Person class, which has *Conductor* and *Passenger* as its subclasses and *Passenger* in turn has *Aggressor* as its subclass. Because passengers and aggressors can have objects with them when traveling by train we can restrict this class by using the existential restriction and the already defined hasObject property. In other words, the passenger and aggressor class has a restriction with property hasObject with a filler of class HumanObject (which contains all the possible objects we defined). In a schematic notation the above would be defined as follows: "$\exists$ *hasObject* **HumanObject**".

The subclasses of *Situation* are also restricted to define them even better. Lets look at the Fighting subclass of *Situation*. We defined that with the following restriction: "$\exists$ *involvedInSituation* (**Aggressor** or **Passenger** or **Conductor** or **Knife** or **Bottle**)". Meaning that if the situation fighting occurs at least one or more of the filler classes could be involved. In this same way we define all the other subclasses of the *Situation* super-class. A note about this, is that these restriction is one possible way of defining a class. In time there could be a revision about the definitions of all the classes in the ontology and changes are possible. As we saw in the previous chapter, ontologies are not always fixed. You can always find room for change and improvements. That's why it is important to be satisfied with the ontology for the type of application it was built. The Situation class needs to be defined as what or who are involved in one of the situations that can occur in the train. To describe that we need to define restrictions.

In table 4.3 we have all the restrictions for the other subclasses of the Situation super-class. A class can have more then 1 restriction (e.g *Graffiti* and *Smoking* subclasses).

Figure 4.9 shows how a restriction defined in table 4.3 looks like in OWL. It's a snippet for the Vandalism class. For the complete ontology in OWL refer to appendix A. The <owl:SomeValueFrom> tag is the OWL representative for the existential quantifier.

Finally we need to define restrictions for the *Scenario* class and its subclasses. For the Scenario class we defined the restriction that this class need to have at least one situation derived from the *Situation* class. This restriction is defined as follows for the Scenario class, $\exists$ *hasSituation* **Situation**. This restriction will be inherited by default by all the subclasses. The subclasses in turn, can specialize and even restrict specifically to one or more of the situations. Let us now look at each of the subclasses of the *Scenario* class. Table 4.4 show all the

Table 4.3: Restrictions of the *Situation* subclasses.

Class	Restrictions
Abandoned_Luggage	∃ *hasObject* **Luggage**
Beggar	∃ *involvedInSituation* **Aggressor**
Drunkard	∃ *involvedInSituation* **Aggressor**
Fainting	∃ *involvedInSituation* **Aggressor** *or* **Passenger**
Fighting	∃ *involvedInSituation* **Aggressor** *or* **Passenger** *or* **Conductor** *or* **Knife** *or* **Bottle**
Graffiti	∃ *involvedInSituation* **Aggressor** *or* **TrainObject** ∃ *hasObject* **Paint**
Phone_Abuse	∃ *involvedInSituation* **Aggressor** ∃ *hasObject* **Phone**
Screaming	∃ *involvedInSituation* **Aggressor** *or* **Passenger**
Smoking	∃ *involvedInSituation* **Aggressor** ∃ *hasObject* **Cigarette**
Theft	∃ *involvedInSituation* **Aggressor**
Vandalism	∃ *involvedInSituation* **Aggressor** *or* **TrainObject**
Vomiting	∃ *involvedInSituation* **Passenger**

restrictions for the scenario class. Again we discuss one of the classes to show what the restrictions represent. For example let us look at the *Annoyance* class. The restriction here is that if the situations *Beggar*, *Theft*, *Smoking*, *Phone_Abuse* and *Drunkard* occur they belong to the scenario *Annoyance*. We have a object property named *hasSituation*. So we define our restriction as follows, ∃ *hasSituation* **Beggar** or **Theft** *or* **Smoking** *or* **Phone_Abuse** **Drunkard**. E.g when at least one of these situations occur, the class Annoyance is triggered as being the scenario of the train domain. A point to note is the *Neutral* class, which occurs when there are no abnormal situations in the train (e.g. everyone behaves (Table 4.4)). This is an example of a cardinality restriction as discussed in the previous section(*hasSituation* = 0).

Table 4.4: Restrictions of the *Scenario* subclasses.

Class	Restrictions
Annoyance	∃ *hasSituation* **Smoking** *or* **Theft** *or* **Phone_Abuse** *or* **Drunkard** *or* **Beggar**
Damage	∃ *hasSituation* **Graffiti** *or* **Vandalism**
Danger	∃ *hasSituation* **Screaming** *or* **Fighting** *or* **Abandoned_Luggage**
Neutral	= *hasSituation* 0
Sickness	∃ *hasSituation* **Vomiting** *or* **Fainting**

```
<owl:Class rdf:ID="Vandalism">
        <rdfs:subClassOf rdf:resource="#Situation"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#involvedInSituation"/>
                <owl:someValuesFrom>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                            <owl:Class rdf:about="#Aggressor"/>
                            <owl:Class rdf:about="#TrainObject"/>
                        </owl:unionOf>
                    </owl:Class>
                </owl:someValuesFrom>
            </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

Figure 4.9: OWL snippet of the complete Vandalism class (including the subclass it belongs to and restriction)

Figure 4.10: Screenshot how to model disjoint classes of Damage class in Protégé.

4.4.2 Disjoint classes

To specify that classes in the ontology cannot occur simultaneously, we make them disjoint from each other. In our modeling program it is easy to make two classes disjoint from each other. All the subclasses of for example *Scenario* super-class are disjoint. Meaning that an instance of Annoyance class cannot also be an instance of the Danger or any of the other subclasses of *Scenario*. In figure 4.10 we show a screenshot how this can be done in Protégé. And in the OWL snippet in figure 4.11 the corresponding OWL code for disjoints is shown.

4.5 An example of reasoning using the aggression ontology

The purpose of the aggression ontology in our case is not to use it as a reasoning mechanism, but to define concepts and their relationships formally, so that for example annotators of train data can use the same vocabulary to annotate the data, and that reasoning systems driven by the ontology will be able to process the data with greater precision. Although it is also possible to make the ontology a reasoning system itself. There are several reasoning

```
<owl:Class rdf:ID="Damage">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasSituation"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <owl:Class rdf:about="#Graffiti"/>
                        <owl:Class rdf:about="#Vandalism"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#Scenario"/>
    <owl:disjointWith rdf:resource="#Annoyance"/>
    <owl:disjointWith rdf:resource="#Danger"/>
    <owl:disjointWith rdf:resource="#Neutral"/>
    <owl:disjointWith rdf:resource="#Sickness"/>
</owl:Class>
```

Figure 4.11: Damage class with disjoint classes.

programs (e.g RacerPro or Pellet) that can do inference with input data using the aggression ontology. In this section we will just briefly give a theoretic example of how such inference would go about. Reasoning in OWL(Description Logic) is based on what is known as the *open world assumption*(OWA). It is frequently referred to as *open world reasoning*(OWR). The *open world assumption* means that we cannot assume something doesn't exist until it is explicitly stated that it does not exist. In other words, because something hasn't been stated to be true, it cannot be assumed to be false – it is assumed that "the knowledge just hasn't been added to the knowledge base". For example in our aggression ontology we have stated that the **Graffiti** and **Vandalism** situations indicates the **Danger**scenario. Because of the open world assumption, until we explicitly say that the **Danger** scenario has *only* these situations, it is assumed(by the reasoner) that a **Danger** scenario could have other situations. This is also a reason to add to what is known as *closure axioms* on the **hasSituation** property. As we saw in previous section this has been done by the quantifier restrictions.

To give an example, lets make instances of *humanObjects*, *Passenger* and *Aggressor*, and use that as the input data. Say the instances are respectively, pocketknife, Henk and Ramon. We relate the pocketknife to Ramon (by objectproperty *hasObject*) and Henk and Ramon are related by *InvolvedInSituation* in our ontology. A reasoner (like RacerPro or Pellet) could infer that given all the input(individuals) and their relations (by object property) that the situation would belong to the **Fighting** class. Which in turn

can lead to the scenario that it is a dangerous scenario (**Danger** class), which in turn can give an alert of some kind to authorities to analyze the scenario and take the necessary actions. The actual reasoning with ontologies can occur in several ways. Data can be added into an SQL or Oracle database and be used to reason with any of the known query languages known for ontologies like SPARQL, OWL-QL , SQWRL, SWRL etc. With any of the query languages the known actual data can be inserted in the query string and used to infer matching output given the data.

Ontology Reasoners are based on OWR as said before, so all relations and properties has to be given explicitly or strange results will occur when the reasoner will try to classify the ontology.

4.6 Summary

Ontology is the theory of objects in terms of the criteria which allow one to distinguish between different types of objects and the relations, dependencies, and properties through which they may be described. The aggression ontology encodes the relational structure of concepts which one can use to describe and reason about aspects of the world. In appendix A the whole aggression ontology as defined in OWL can be found. In this chapter we described how we designed our ontology. Also we established what the scope of this ontology was. As described previously we designed this ontology to just give a better understanding of our train domain and all the concepts in this domain. It clearly give a complete overview of all the aspects and relationships between concepts that take place in the train domain. The design started with the important concepts in our ontology, which led to the class hierarchy. Object properties (slots) were implemented which relate different concepts(classes) to each other. The relation between concepts is achieved by describing the domain and range classes of an object property. Restriction in classes were implemented to restrict the individuals that belong to a class. This is useful when the ontology will be used in a reasoning of inference application. Then individuals can be inferred to their appropriate class.

It should be noted, that what is accomplished by using ontologies cannot be accomplished by using purely syntactic languages such as XML schema. As XML schema only specify the structure of objects (i.e. their composition). While ontologies capture the semantic meaning i.e. knowledge of how the classes of data objects relate to each other.

An ontology can always be improved as new knowledge or improved knowledge about the domain becomes available. So in time ontologies will need to change. Even in our aggression ontology, numerous changes have been made over time. But if the ontology satisfies the scope it is made for then we should leave it as is, until a next major change in the scope. Although our ontology in itself was

not designed to reason about the train data, we tried to give an example in the last section of this chapter how ontology reasoners would try to classify and infer with the data to come to a conclusion. The example is a simple one but more complex reasoning capabilities are possible with ontologies. Now we can translate this ontology into a Bayesian network to reason with different situations occurring in the train. The next chapter will describe the Bayesian network based on this ontology.

Chapter 5

Bayesian Network: Background Theory

We have to find an appropriate tool able to represent various sources of uncertain information that describe our problem, and to join them into an inference system. This tool should be able to represent a degree of uncertainty, i.e. the probability of the objects or events, and relations that exists between these objects, and events. We can deal with uncertainty in two ways: extensionally and intentionally. Extensional systems (also called rule-based systems) are generally computationally efficient but their uncertainty measures are semantically weak. Knowledge rules can be used as described in [19]. Knowledge rules are described on a high level context-sensitive grammar. On the contrary, intentional systems are generally computationally expensive and semantically strong. In probability reasoning, random variables are used to represent event and/or objects in the world. By assigning values to these random variables, we can model the current state of the world (in our case the train environment) and weight the states according to the joint probabilities. If we consider above mentioned probability and uncertainty, this will lead us to the use of Bayesian Networks (BNs). Bayesian Networks bring the most appropriate representation of relative influences among real world facts.

The goal of this chapter is to give background information about what Bayesian Networks(BN)are. How it is structured as well as how inference and parameter learning can be done on BNs.

5.1 Bayesian Networks

A Bayesian Network (BN), also known as a belief network or a Bayesian belief network, is a graphical model for probabilistic relationships among a set of variables [23]. For over a decade, expert systems use BNs in domains where uncertainty plays an important role. Nowadays, BNs appear in a wide range of

diagnostic medical systems, fraud detection systems, missile detection systems, etc.

Why are BNs so interesting? BNs have a couple of properties that make them so popular and suitable to use. The five most important properties are, in no particular order, the following: (1) BNs can handle incomplete data sets; (2) BNs allow one to learn about relationships between variables; (3) it is possible to combine expert knowledge and data into a BN; (4) because of the use of Bayesian methods, the over-fitting of data during learning can be avoided relatively easy; and (5) BNs are able to model causal relationships between variables. BNs have a qualitative and a quantitative component. The qualitative component is represented by the network structure and the quantitative component is expressed by the assignment of the conditional probability distributions to the nodes. Before we discuss the network structure and conditional probability distributions, first Bayes' Theorem is presented.

5.1.1 Bayesian probability theory

The main building block of BN theory is *Bayes' Theorem*. This theorem is stated as follows:

$$p(X|Y) = \frac{p(Y|X) \cdot p(X)}{p(Y)}, \tag{5.1}$$

where:

- $p(X|Y)$ is the posterior probability of the hypothesis X, given the data Y,
- $p(Y|X)$ is the probability of the data Y, given the hypothesis X, or the likelihood of the data,
- p(X) is the prior probability of the hypothesis X, and
- p(Y) is the prior probability of the data Y, or the evidence.

Equation 5.1 states that by observing evidence, the prior probability of the hypothesis changes to a posterior probability.

Two other rules are important in BNs. The first one is the *expansion rule*. Consider the situation where X and Y are random variables with k possible outcomes:

$$\begin{aligned} p(X) &= p(X|y^{k=1}) \cdot p(y^{k=1}) + p(X|y^{k=2}) \cdot p(y^{k=2}) + \cdots + p(X|y^{k=k}) \cdot p(y^{k=k}) \\ &= \sum_{Y} p(X|Y).p(Y). \end{aligned} \tag{5.2}$$

Applying the expansion rule means that we can introduce variables on the right-hand side of the equation, as long as we sum over all their possible values.

This concept is also known as the *marginal probability* of X, meaning that only the probability of one variable (X) is important and all information about other variables (Y) is ignored. The second rule is the *chain rule*. This rule is derived by writing Bayes' Theorem in the following form, which is called the *product rule*:

$$\begin{aligned} p(X,Y) &= p(X|Y)\cdot p(Y) \\ &= p(Y|X)\cdot p(X). \end{aligned} \tag{5.3}$$

Successive application of the product rule yields the chain rule:

$$\begin{aligned} p(X_1,\cdots,X_n) &= p(X_1,\cdots,X_{n-1})\cdot p(X_n|X_1,\cdots,X_{n-1}) \\ &= p(X_1,\cdots,X_n-2)\cdot p(X_{n-1}|X_1,\cdots,X_{n-2})\cdot p(X_n|X_1,\cdots,X_{n-1}) \\ &\cdots\cdots \\ &= p(X_1)p(X_2|X_1)\cdots p(X_n|X_1,\cdots,X_{n-1}) \\ &= p(X_1)\prod_{i=2}^{n} p(X_i|X_1,\cdots,X_{i-1}). \end{aligned} \tag{5.4}$$

For every X_i, there may be a subset of (X_1,$\cdots$,X_{i-1}) such that X_i and the subset are conditionally independent. This means that this subset can be left out of the original set (X_1,$\cdots$,X_{i-1}). When the subset is empty, X_i is conditionally dependent on all variables in (X_1, $\cdots$,X_{i-1}). In simpler terms, based on the structure of the network it is sometimes possible to remove variables in the equation on the right hand side. If a variable A is conditioned on a variable B and C, so $P(A|B;C)$, and A is conditionally independent of B, $P(A|B;C)$ can be reduced to $P(A|C)$. This lowers the number of parameters required to specify the JPD(Joint Probability Distribution) by exploiting independencies among domain variables.

So a BN is basically a compact representation of a joint probability distribution (JPD). The JPD is specified by means of local probability distributions associated with nodes (variables). In case of discrete variables (this is what we will focus on in this thesis), the local probability distributions are encoded in the form of prior probabilities over those nodes that have no parents in the graph, and conditional probability tables (CPTs) for all other nodes, see for an example Figure 5.1. A CPT is a set of conditional probability distributions that define a probability distribution over the child variable given all combinations of values of the parents nodes.

5.1.2 Network Structure

The qualitative part of the BN is represented by its structure. A BN is a directed, acyclic graph (DAG) where the nodes represent the set of random variables and the directed arcs connect pairs of nodes. We have already seen

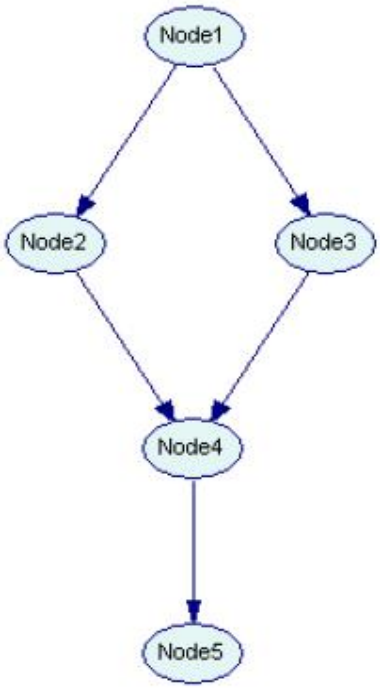


Figure 5.1: Example of the graphical part of a Bayesian Network.

two examples of BN structures in the figures 5.1. The nodes in the BN represent *discrete*[1] random variables. If an arc points from X_1 to X_2 then X_1 is a *parent* of X_2 and X_2 is a *child* of X1. A parent directly influences its children. Furthermore, every node has its own local probability distribution. All these components together form the joint probability distribution (JPD) of the BN. The joint probability distribution of **X** follows from applying equation 5.4 on the network structure. The parents of X are denoted by **Pa**(X):

$$p(X_1, \cdots, X_n) = \prod_{i=1}^{n} p(X_i|Pa(X_i)). \tag{5.5}$$

$p(X_i|Pa(X_i))$ is called the *local probability distribution* (LPD). The process of breaking up the joint probability distribution into local probability distributions is called *factorization*, which results in an efficient representation that supports fast inference. This is a property of BNs that forms a major contribution to its success. To come back to the restriction for a BN to be a DAG, this follows from equation 5.5. To see this, imagine the network in figure 5.2. For this

[1]The theory can be easily extended to handle continuous random variables with arbitrary probability distributions.

network, the JPD can be calculated by:

$$\begin{aligned} p(X_1, X_2, X_3) &= p(X_1|Pa(X_1)) \cdot p(X_2|Pa(X_2)) \cdot p(X_3|Pa(X_3)) \\ &= p(X_1|X_3) \cdot p(X_2|X_1) \cdot p(X_3|X_2). \end{aligned} \tag{5.6}$$

It is easy to see that the resulting calculation can never be written back to the original chain rule (equation 5.4), which in its turn was a direct result of successive application of the product rule form of Bayes' Theorem. In short, allowing cycles in the network is not consistent with Bayes' theorem!

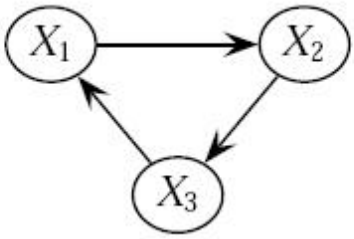


Figure 5.2: A Network with a Cycle.

5.1.3 Conditional Independence

Conditional independence is an important concept in Bayesian Networks and it can be specified by:

- A node is conditionally independent of its non-descendants, given its parents.

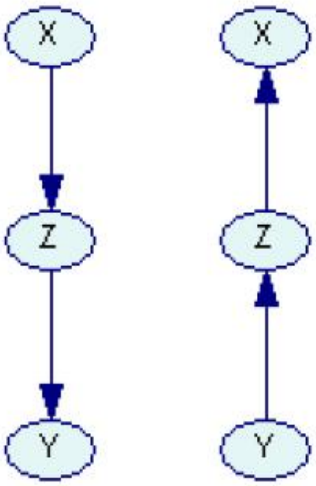


Figure 5.3: Serial Connection in both Directions.

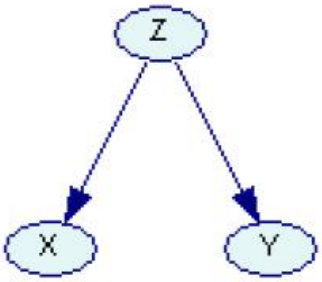


Figure 5.4: A diverging connection.

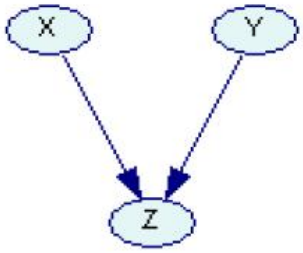


Figure 5.5: A converging connection.

- A node is conditionally independent of all other nodes in the network, given its parents, children, and children's parents. This is called the Markov blanket.

These two specifications are equivalent. A more general topological criterion is d-separation. It can be used to decide whether a set of nodes **X** is independent of another set **Y**, given a third set **Z**. It is more complicated than the two previously mentioned methods and it works as follows.

Two nodes, X and Y are d-separated if and only if for every path between them, there is an intermediate variable Z such that:

- The connection is serial or diverging and Z is known. A connection is serial if there is a path from X to Y or Y to X through Z, see also Figure 5.3. A connection is diverging if node Z has an arc to X and an arc to Y , see also Figure 5.4.
- The connection is converging and neither Z nor any descendant of Z is known. A connection is converging if there is an arc from X and an arc from Y to Z, see also Figure 5.5.

Node X and Y are d-connected by node Z, if they are not d-separated. If two nodes are d-separated they are independent. This is very useful for inference at which we will take a look in the next section.

5.2 Inference

Inference is a basic task in BNs and is used to compute the posterior probability distribution for a query variable $Q \in \mathbf{Q}$, given the set of evidence (observed) variables $\mathbf{E}$. There is a possible third set of (hidden) variables $\mathbf{H}$ that are neither query variables nor observed variables. The complete set of variables of a query is $\mathbf{X} = \{Q\} \bigcup \mathbf{E} \bigcup \mathbf{H}$. The posterior probability can be calculated in this way:

$$P(Q|\mathbf{E}) = \frac{P(Q,\mathbf{E})}{P(\mathbf{E})} = \alpha P(Q,\mathbf{E}) = \alpha \sum_{H} P(Q,\mathbf{E},\mathbf{H}),$$

where α is a normalizing constant equal to $\frac{1}{P(\mathbf{E})}$.Informally, the equation says that a query can be answered using a Bayesian Network by summing out the variables in $\mathbf{H}$ and dividing that by the probability of the evidence.

Speeding up inference is achieved by manipulating the sum in the equation in a clever way. One easy improvement is to move the constant terms out of the sum. A second improvement is to compute repeating values only once. There are values that do not even need to be computed: every variable that is not an ancestor of a query variable or evidence variable is irrelevant to the query. Clustering algorithms (also known as joint tree algorithms) [27] are very useful when someone is not interested in only one query variable, but if, for example, the values of all the variables in the network have to be computed. The idea is to join individual nodes of the network to form cluster nodes in such a way that the resulting network is a polytree. When this network is in polytree form, a special purpose inference algorithm is applied that can compute the values of all the nodes in the network in $O(n)$ time. But the NP-hardness does not disappear: the construction of the polytree requires exponential time and space if the network is difficult enough.

Since inference in BNs is NP-hard, approximate algorithms have been developed to handle large multiply connected networks. There is a variety of approximate algorithms:

- Direct sampling methods.
- Rejection sampling.
- Likelihood weighting [10].
- Markov Chain Monte Carlo (MCMC) [36].

The idea with direct sampling methods is to generate evidence for each variable following a partial order defined by the structure of the graph. The probability distribution from which the value is sampled is conditioned on the evidence of the parents of the node.

Rejection sampling can be used to compute conditional probabilities. Samples are generated by using the prior distribution of the network and samples are removed that are not consistent with the network. Finally, we count the occurrences of the state of the query variable to estimate the probability.

Likelihood weighting is an improvement over rejection sampling in the sense that it only generates samples that are consistent with the network.

MCMC does not generate each event from scratch, but jumps from state to state. The next state is generated by randomly sampling a value for one of the non-evidence variables X_i, conditioned on the current values of the variables in the Markov blanket of X_i. The idea is that in the long run the number of visits to each state is exactly proportional to its posterior probability.

5.3 Types of Explanations in Bayesian Networks

Literature on the topic of explanations in Bayesian Networks distinguishes three kinds of explanations [26]. The first one is called *abduction*. Abduction is the process of determining the most probable values of the unobserved variables in a Bayesian Network. Such a configuration is usually referred to as an MPE (Most Probable Explanation) and can contain every unobserved variable, in which case it is called *total* abduction, or it can contain only a subset of the unobserved variables, in which case it is called *partial* abduction. Abduction involves maximizing the probability of a set of unobserved variables given one or more findings. It is also possible to generate a set of MPEs, for example the five configurations with the five highest probabilities.

The other two kinds of explanations are *static* and *dynamic* explanations. A *static* explanation only considers the information that is contained in the Bayesian Network model, i.e., without any reasoning being done. Another way of putting it is that it offers explanations of the assumptions of the model. A static explanation could, for example, make the independence statements contained in a model explicit, or it could describe the prior probability of variables.

A *dynamic* explanation, on the other hand, is an explanation of the reasoning process in a Bayesian Network. So, given one or more findings and a variable of interest, a *dynamic* explanation tries to give the user insight into the process that caused the variable of interest to be affected in the way that it has. More specifically, it tries to explain the changes in the posterior probability of the variable of interest with respect to the findings. This type of explanation can be viewed as trying to answer the questions: "What were the most influential findings?" and "Why is a certain finding influential?". A finding is influential when it affects the posterior probability of the variable of interest in either a positive or a negative way.

Within dynamic explanations another distinction can be made. There is a difference between *micro* and *macro* explanations [40]. A *micro* explanation tries to justify the variations of the probability distribution of a certain node, while a *macro* explanation tries to make the main lines of reasoning from findings to variable of interest clear to the user and therefore considers a bigger part of the model.

An explanation should be presented in a way that is effective, convenient, as well as easily accessible. A distinction that can be made in this respect is that between *verbal* and *graphical* explanations. A *verbal* explanation could be, for example: "Variable A is dependent on variable B, but given variable C they are independent", or "State zero is somewhat more likely than state one".

A *graphical* explanation uses graphical means to communicate an explanation. The most obvious and basic explanation of this type is the visualization of the network structure. If the user has enough knowledge about Bayesian Networks, he can deduce the dependencies and independencies between the variables in the modeled domain from this view. Another example is to display the probabilities of the various states of a variable using graphical bars that range from zero to one hundred percent. Some of the BN software discussed in the next section use graphical explanations.

5.4 Current Software for Bayesian Networks

In this section we present a general overview of software available to assist us in working with Bayesian Networks. In this chapter only the software that could be investigated directly is covered. Investigating this software gives insight on which features are useful, which are currently offered and which are currently missing.

5.4.1 The Bayes Net Toolbox for Matlab

Until the introduction of the Bayes Net Toolbox for Matlab (BNT) [33], the field lacked a free general purpose software library that was able to handle many different variants of graphical models, inference and learning techniques. The BNT is an attempt to build such a free, open-source, and easy-to-extend library that can be used for research purposes.

The author chose to implement the library in Matlab because of the ease with which it can handle Gaussian random variables. Matlab has various advantages and disadvantages, the main disadvantage being that it is terribly slow.

In the BNT, BNs are represented as a structure containing the graph, the CPDs and a few other pieces of information. The BNT offers a variety of inference algorithms, each of which makes different trade offs between accuracy,

generality, simplicity, speed, etc. All inference methods have the same API, so they can be easily interchanged. The conditional probabilities of the defined variables can be continuous or discrete. Parameter and structure learning are supported as well.

The toolbox was the first of its kind and set a standard for (D)BN libraries. It is still widely in use today, because the code is relatively easy to extend and well documented and the library has by far the most functionality of all software currently available. However, much functionality still needs to be added to make it a real general purpose tool, such as tree-structured CPDs, Bayesian modeling, online inference and learning, prediction, more approximate inference algorithms, and support for nonDAG graphical models. Also, the scripting part that is needed to implement a model in the BNT can be really cumbersome. Added to that, BNT does not have support for standard BN file formats, which cancels the possibility to export and/or import BN models from/to other packages. The BNT is released as open-source software and can be downloaded from http://bnt.sourceforge.net. A GUI is currently in development.

5.4.2 The graphical models toolkit

The graphical models toolkit (GMTK) [4] is a freely-available and open-source toolkit written in C++ that is specialized in developing DBN-based automatic speech recognition (ASR) systems. The GMTK has a number of features that can be used for a large set of statistical models. These features include several inference techniques, continuous observation variables and discrete dependency specifications between discrete variables. The DBN model needs to be specified in a special purpose language. In this language, a DBN is specified as a template that contains several time-slices. The collection of time-slices is unrolled over time to create an unrolled DBN. The time-slices in the template are divided into a set of prologue, repeating and epilogue time-slices and only the repeating frames are copied over time. This approach to modeling DBNs has much expressive power, but it is also a lot of work to specify a DBN model.

The GMTK is a promising library. To become really useful, a couple of disadvantages need to be tackled. Its main disadvantages are that it is not a general purpose toolkit, because it specializes in ASR and it therefore lacks much functionality that is useful for other applications. Furthermore, the documentation is far from complete, making it difficult for a user that is not closely involved in the development of the software to understand the toolkit. Finally, although the author of the toolkit states that it is open-source, the website (http://ssli.ee. washington.edu/ bilmes/gmtk) still only offers a binary version.

5.4.3 SMILE

The Structural Modeling, Inference, and Learning Engine is a fully platform independent library of C++ classes that implements graphical probabilistic and decision-theoretic models and is suitable for direct inclusion in intelligent systems. The individual classes defined in the Application Program Interface (API) of SMILE enables the user to create, edit, save and load graphical models, and use them for probabilistic reasoning and decision making under uncertainty. To be able to access the SMILE library from a number of other programming languages, wrappers exist for ActiveX, Java, and .NET. SMILE was first released in 1997 and has been thoroughly used in the field since. Its main features are:

- Platform independent; versions available for Windows, Unix (Solaris), Linux, Mac, PocketPC, etc.
- Wrapper available for use with .NET framework. Compatible with all .NET languages. May be used to create web-based applications of BNs.
- Thorough and complete documentation.
- Robust and running successfully in the field since 1997.
- Responsive development team support.

5.4.4 GeNIe

The Graphical Network Interface is the graphical user interface to the SMILE library. It is implemented in C++ and makes heavy use of the Microsoft Foundation Classes. Its emphasis is on accessibility and friendliness of the user interface, making creation of decision theoretic models intuitive using a graphical click-and-drop interface approach. It is a versatile and user-friendly environment for building graphical decision models and performing diagnosis. Its primary features are:

- Graphical editor to create and modify models.
- Supports nodes with General, Noisy-OR/MAX and Noisy-AND/MIN probability distributions.
- Functionality to cut and paste parts from/to different BNs.
- Complete integration with Microsoft Excel, cut and paste data into internal spreadsheet view of GeNIe.
- Cross compatibility with other software. Supports all major file types (e.g. Hugin, Netica, Ergo).

- Support for handling observation costs of nodes.
- Support for diagnostic case management.

Nowadays, the combination of SMILE and GeNIe has several thousand users worldwide. Applications based on the software range from heavy-duty industrial software to academic research projects. Some examples are: battle damage assessment, group decision support models for regional conflict detection, intelligent tutoring systems, medical diagnosis and therapy planning, diagnosis of diesel locomotives, airplanes, and production processes of integrated circuits. SMILE and GeNIe are also used for teaching statistics and decision-theoretic methods at several universities.

5.4.5 BayesiaLAB

BayesiaLab is the BN software toolkit developed by the French company Bayesia (http://www. bayesia.com). BayesiaLAB has two modi: a modeling mode for designing the BN and a validation mode for inference. In the modeling mode, it is possible to add temporal arcs to a BN to indicate that the parent node of the two nodes is in the previous time-slice. The user needs to specify the initial state and the temporal probabilities of the nodes. After specification, the validation mode can be selected to follow the changes of the system over time by browsing between time steps (only forward steps are supported) or by setting the number of steps and then plotting a graph of the variables. The DBN does not unroll graphically, only the values change. It is possible to set evidence by hand or by importing a data file. Of the inference methods known, only filtering and prediction are possible.

5.4.6 Netica

Netica is a product of Norsys (http://www.norsys.com), which is located in Canada. In Netica, the user designs the (D)BN and then compiles it, after which inference is possible. Compiling the (D)BN basically means that it is converted to a junction tree representation. When designing a DBN, temporal arcs can be added between nodes. The difference with BayesiaLAB is that we explicitly define the temporal relationship (called a time-delay in Netica) of two nodes in one of the nodes instead of the implicit approach of BayesiaLAB. When the DBN has nodes with temporal relations to itself (which is usually the case), the DBN definition contains loops. Of course, this is only valid in the definition of the DBN; the unrolled version should not contain loops. After defining the DBN, it can be unrolled for time slices and compiled. Inference can be performed on this unrolled and compiled BN. Most inference methods known are supported. We can enter evidence by hand or by importing a data file(Case file).

5.5 Summary

A Bayesian Network provides a complete description of the domain. Every entry in the full joint probability distribution can be calculated from the information in the network(evidence). This chapter presented the theoretical foundation of Bayesian reasoning on which the remainder of this thesis relies. It presented a global overview of BN theory and techniques that is vital for a general understanding of the rest of this thesis, such as d-separation, inference, and learning. Furthermore, a short comparison of BN modeling tools was given, and in-depth reference material on different aspects of Bayesian Networks was presented. As such, this chapter can serve as a good starting point for readers that are trying to get familiar with BN theory and techniques.

In the next chapter the BN formalism as presented in this chapter will be extended and implemented for the goal of this thesis and that is to transform the ontology designed in chapter 4 into a Bayesian Network to reason with the input data and and give a indication of the threat(scenario) in the train compartment. This will be done with the means of a case study with actual footage shot in a real train environment which will be presented in chapter 7.

Chapter 6

Design of the Bayesian Network

In this chapter we will present the bayesian network based on the defined concepts and relations between the concepts as designed in the aggression ontology.

6.1 Overview

There are two ways in which one can understand the semantics of Bayesian Networks. The first is to see the network as a representation of the joint probability distribution. The second view it as an encoding of a collection of conditional independence statements. The two views are equivalent, but the first turns out to be helpful in understanding how to construct networks, whereas the second is helpful in designing inference procedures. The goal in our case is to design a robust bayesian network based on the aggression ontology as a basis for the semantic meaning of the nodes in this network. In the next section we will give a theoretical explanation of the global design of the Bayesian Network. We will first discuss and present the main high-level features of which the different situations and scenarios that may occur in a train compartment consist of. A note on the side; in this thesis we mainly concentrated on the high-level features in a train compartment. It is possible we will explain some low-level features, but that is merely an exception or to give the reader a better explanation of the model.

6.2 Global design

This section will provide the global design of our constructed aggression Bayesian Network. The goal of the bayesian network is to let is infer, given the input features, what the expected scenario is in the train compartment. So the

output nodes will be the 5 scenarios presented in chapter 4. To sum up the output nodes will be : *Damage, Danger, Annoyance, Sickness and Neutral.* These are the scenarios and their consisted situations that were utilized throughout the development of the bayesian network and this thesis. The 12 situations that we covered in this thesis are: *Abandoned_Luggage, Beggar, Drunkard, Fainting, Fighting, Graffiti, Phone_Abuse, Screaming, Smoking, Theft, Vandalism, Vomiting.* In the Bayesian Network each situation will be an node in the structure. In our ontology design we covered which situations represented its corresponding scenario class. So in our bayesian network, there will be connections between a situation- and its corresponding scenario node.

6.2.1 Features

We will first give an overview of all the features that implicate the given twelve situations. These features have been extracted from actual footage shot in a real train environment. Features are actually a short description was the scene playing in the train at a given time. Such a feature will have keywords as described in our aggression ontology as a annotation helper. Table 6.1 give an overview of all the features extracted from the footage for our situations.

Table 6.1: This table lists the derived features.

Features
1. Person in unconscious.
2. Person is providing help.
3. Agitated Voice.
4. People are talking in panicky manner.
5. Person is puking.
6. Person is shouting/screaming.
7. Person falls to the floor.
8. Person is lying on the walkpath.
9. Person calling on a phone.
10. People are complaining/annoyed.
11. Person is talking loud.
12. Person is invading personal space of another person.
13. People are in a quarrel.
14. Person is waving with a knife.
15. Person is painting on the train interior.
16. Person is acting in a suspiciously.
17. Person is damaging train interior.
18. Luggage lying around with no person in sight.
19. Person accusing other person of pick-pocketing.
20. Person moving around in a drunken manner.
21. Person has a beer in his hand.
22. Person talks rubbish to others.
23. Person is lighting a cigarette.
24. Person moves around making begging gestures.

The description of the features are self-explaining. We've limited the number of features somewhat because the network would become very big and one would lost the total overview. On the other hand, the more information(nodes) is available the more robust a Bayesian Network is. We went for a trade-off between both. Just enough features to make both a robust network and not leave any essential information out. We mixed in our features table some low-level features as well just to give an example of how they can help instantiate the higher-level features. These higher-level features represent combinations of lower-level features and serve to add an additional layer of semantics to the underlying features. In such a manner, a set of low-level features that may separately bear little meaning can be combined into a higher-level feature that does associate a meaningful interpretation to its underlying sub-features. As noted before in this thesis we mostly concentrated on the higher-level part of the semantics, but for a more accurate picture we mixed some essential low-level features into the bayesian network. To give an example of the a high-level feature : "People are in a quarrel" which is number 13 in table 6.1. This feature can be comprised of the lower-level features: "Person moving towards other person" and "People pushing/shoving each other".

6.2.2 Composition of the different situations

The features extracted from the footage will now be combined in the Bayesian Network. We will first give a overview of which features are comprised of for the twelve situations. Table 6.2 give the situations and their corresponding features that when active in the scene will indicate the possibility of the given situation. For simplicity of the table we only stated the feature number as given in table 6.1. Another way to look at this is that the features give a better understanding of the definition of the situation.

Table 6.2: Situation definition.

Situation	**Feature number**
Abandoned_Luggage	18
Beggar	10,24
Drunkard	20,21,22
Fainting	1,2,4,7,8
Fighting	12,13,14
Graffiti	15,16
Phone_Abuse	9,10,11
Screaming	3,6
Smoking	10,24
Theft	11,13,19
Vandalism	16,17
Vomiting	2,4,5

Now that we have the established the global design of all the features related to the situations, and how the situations are related to scenarios, we can now implement the Bayesian Network using the modeling software Genie. The next section will cover the complete Bayesian Network.

6.3 Implementation

This section aims to describe the implementation details of the constructed Bayesian Network. In order to achieve this, the following will start off with the presentation discussion of the networks final structural organization. Next, the technical details will provided, primarily focusing on the Bayesian Networks conditional probability tables.

6.3.1 Bayesian Model

This section we will present the final layout of the Bayesian Network. In previous chapter we established that the five scenario nodes are the final output nodes of the Bayesian Network. They will give an estimate probability of each scenario given the input feature nodes. One discussion for these output nodes is how many states to attach to them. We could have attached 2,3 or 5 level states, but to keep it simple and also due to the idea that the final output nodes should give just the probability of that scenario occurs at that given moment, we decided to just attach one state named which will represent the probability. Now if we compare all the 5 probabilities, the one with the highest is the scenario most likely to occur given the input data from the high-feature nodes. Another general approach we used for the overall network is to keep the conditional probability tables(CPTs) in proportions. When deciding on the arrangement of the network, it was of primary importance to reduce the number of incoming connections for each node to at the most 4, in order to prevent the conditional probability tables from growing too large.

The next task was to define the feature nodes. As described in the previous section we made a list of features which counted 24 different features. These will all be modeled as a node in our network. The concept of higher-level features was revisited several times by us, coupled with their implementation in the Bayesian network. While, in general, successive nodes in a Bayesian Network have their connection defined through means of a conditional probability table (CPT), we could decide to have the higher-level feature nodes implemented in a different fashion. Analog to the manner in which general rule-based expert systems activates input data, the Bayesian Network should be capable of deterministically enabling higher-level features when necessary. For our Bayesian Network we wanted to have this capability also. Therefore, instead of relying on a traditional probability-based network node, higher-level

features are translated to their Bayesian equivalent using deterministic network nodes. Instead of having probabilities dictate the value of such nodes, their outcomes are precisely defined for every possible set of values of previous nodes. As an example, this would effectively allow the network to activate higher-level feature C, if lower-level features A and B are activated. Deterministic nodes, usually drawn as double-circles in the Genie modeling software we used, represent either constant values or values that are algebraically determined from the states of their parents. In other words, if the values of their parents are known, then the value of a deterministic node is also known with certainty. Deterministic nodes are quantified similarly to Chance nodes. The only difference is that their probability tables contain all zeros or ones (note that there is no uncertainty about the outcome of a deterministic node once all its parents are known). To have this as a possibility is very efficient as we can actually switch features on and off if respectively occur or not. This could have been a good reason to choose for deterministic nodes to implement our high-level nodes, on the other hand if we take a look at a real-world it is often not as black and white if a feature is really active. So the idea is to have the flexibility of probabilities still with us, to give the chance that a feature might be active or not. Thus our choice to go for a probability approach for these nodes rather then deterministic nodes.

The last part of nodes to describe for our network would be the intermediate nodes, which would be the 12 situations which we also covered in previous section and which can be found in our ontology too. The situations are just a 1-1 map from the ontology. We modeled our train environment to contain these situations. In addition these situations could be extended in the future, but we maintained the core situations and used them in our Bayesian Network and ontology. The situation nodes are probabilistic nodes and have the high-level feature nodes are their parents and they are the parent nodes of the scenario output nodes. Specific high-level features are connected to their corresponding situation node given as in Table 6.2. Equivalently the situation nodes are connected to their corresponding scenario node as given in the aggression ontology (Table 4.1) in chapter 4.

We will now present the network structure of our Bayesian Network. Figure 6.1 presents our complete Bayesian Network. We colored the nodes to give a clear overview of the different groupings of the different nodes. Due to the high number of nodes (mostly the high-level feature nodes) it looks somewhat chaotic, but it has been carefully designed to give a clear picture of which high-level features implicate corresponding situation nodes. In the next section we will discuss the technical details of the network a bit further especially how we calculated the conditional probability tables of the chance nodes. The high-level feature nodes have 2 states namely Inactive and Active and have a probability attached to them.

6.3.2 Technical Details

This section presents the technical details of the constructed Bayesian Network designed in the previous section. In practice, this entails the presentation and discussion of the conditional probability tables and their contained values. Before providing the actual tables, a brief outline of the approach taken in order to determine the proper probability values is presented. Probabilities can be determined in several ways. One possibility is that it can be determined by a human expert. In a real-world application this would indicate a expert of the domain of the problem at hand.

Other possibilities are to if there is a lot of empirical data available to use that and make the network learn its conditional probability tables using complex learning algorithms. As the latter is not the case, we decided for our research and educational purposes on taking the role of the human expert and decide on the conditional probability tables' contents. Because in our design we took into account to limit the number of input arcs to each of the change nodes, we managed to keep the conditional probability tables at a manageable size. However, it still was a tedious job to determine the CPTs. On the other hand, once in the future more empirical data becomes available due to putting it in action in the real-world, the initial probability tables designed by us can easily be updated.

Lets start with the scenario output nodes. These are the *Neutral, Danger, Damage, Annoyance and Sickness* nodes. We see that the normal node has all the input arcs from all the situation nodes, which makes its CPT very large, on the other hand because the neutral(normal) node only indicates that the scenario is normal if all the situation input nodes have a high low probability for their node. Remember that the situation nodes have 2 states per node(low or high) which have a probability attached to them. So if all the input nodes have low probability we determined that the scenario is neutral in 90% of the cases(so in 10% we are wrong). And if all the probability of all the situations are high then there is a 5% chance the scenario is neutral. And for all other possibilities, meaning 1 or more situations have a high probability of being active, we assign a chance of 20% that the scenario is neutral(80% chance we are wrong). Table 6.3 show just the beginning and the end of the conditional probability table. Because the table is very big and the middle part is just the same we just show the important piece.

For the other 4 scenario nodes we determined the CPTs by means of a aggression scale survey we did during our literature survey. In this survey we asked 31 students who commute daily by means of train to rate different situations(38 questions) on a aggression scale from 1 to 5 (safe-neutral-danger) and also if they actively or passively experience it while occurring. The complete survey and survey analysis has been attached in appendix B. So we

Table 6.3: Partial Conditional Probability table for Neutral.

Neutral												
Abandoned_luggage	Low						High					
Beggar	Low						High					
Drunkard	Low						High					
Fainting	Low						High					
Fighting	Low						High					
Graffiti	Low						High					
Phone_Abbuse	Low						High					
Smoking	Low						High					
Theft	Low						High					
Vandalism	Low		High				Low				High	
Vomiting	L	H	L	H	...	...	...	...	L	H	L	H
Low	0.1	0.8	0.8	0.8	...	...	...	...	0.8	0.8	0.8	0.95
High	0.9	0.2	0.2	0.2	...	...	...	...	0.2	0.2	0.2	0.05

Table 6.4: Conditional Probability table for Danger.

Danger				
Abandoned_luggage	Low		High	
Fighting	Low	High	Low	High
Low	0.9	0.25	0.35	0.05
High	0.1	0.75	0.65	0.95

will use the knowledge gathered from other experts in the field, in this case the commuters to determine the CPTs of the remaining 4 scenario nodes. To do this we choose the question in our survey that match the 4 remaining scenarios and see how the response was. We then use that response to give the initial CPTs for these nodes. We need to point out that this is an subjective data. We use this approach because we lack real empirical data and a initial CPTs are required in order for the Bayesian Network to be effective. So after a lot of tweaking on the different CPTs of both the Scenario- and Situation nodes we will present the final initial CPTs we came up with. Note that for some tables we just gave a partial table, because of the large number of incoming node arcs which makes the table very large to display properly. To view the complete CPT refer to the Genie model of the Bayesian Network.

Below we present the conditional probability tables for the different situation nodes (Tables 6.8 - 6.17). For simplicity we left out 2 nodes because their CPTs would not fit nicely as they are too big. Again we refer to the Bayesian Network file of Genie for the complete overview of all the CPTs. In the tables (6.8 - 6.17) we mentioned the high-level features and added for convenience the number in front of it as we mentioned it in table 6.1.

Table 6.5: Conditional Probability table for Damage.

Damage				
Graffiti	Low		High	
Vandalism	Low	High	Low	High
Low	0.9	0.1	0.1	0.1
High	0.1	0.9	0.9	0.9

Table 6.6: Conditional Probability table for Damage.

Sickness				
Fainting	Low		High	
Vomiting	Low	High	Low	High
Low	0.95	0.15	0.15	0.05
High	0.05	0.85	0.85	0.95

Table 6.7: Partial Conditional Probability table for Annoyance.

Annoyance												
Beggar	Low						High					
Phone_Abuse	Low						High					
Theft	Low						High					
Smoking	Low		High						Low		High	
Drunkard	L	H	L	H	...	...	...	...	L	H	L	H
Low	0.95	0.9	0.85	0.75	...	...	...	...	0.3	0.15	0.1	0.05
High	0.05	0.1	0.15	0.25	...	...	...	...	0.7	0.85	0.9	0.95

Table 6.8: Conditional Probability table for Beggar node.

Beggar				
24.Person is moving around making begging gestures	Inactive		Active	
10.People are complaining	Inactive	Active	Inactive	Active
Low	0.9	0.5	0.2	0.05
High	0.1	0.5	0.8	0.95

Table 6.9: Conditional Probability table for Graffiti node.

Graffiti				
16.Person is acting suspiciously	Inactive		Active	
15.Person is painting on train interior	Inactive	Active	Inactive	Active
Low	0.95	0.1	0.5	0.05
High	0.05	0.9	0.5	0.95

Table 6.10: Conditional Probability table for Screaming node.

Screaming				
6.Person is shouting	Inactive		Active	
3.Agitated voice	Inactive	Active	Inactive	Active
Low	1	0.4	0.4	0
High	0	0.6	0.6	1

Table 6.11: Conditional Probability table for Smoking node.

Smoking				
23.Person is lighting a cigarette	Inactive		Active	
10.People are complaining	Inactive	Active	Inactive	Active
Low	0.9	0.5	0.1	0.05
High	0.1	0.5	0.9	0.95

Table 6.12: Conditional Probability table for Vandalism node.

Vandalism				
16.Person is acting suspiciously	Inactive		Active	
17.Person is damaging train interior	Inactive	Active	Inactive	Active
Low	0.9	0.5	0.1	0.05
High	0.1	0.5	0.9	0.95

Table 6.13: Conditional Probability table for Phone_Abuse node.

Phone_Abuse								
9.Person is calling on the phone	Inactive				Active			
11.Person is talking loud	Inactive		Active		Inactive		Active	
10. People are complaining	I	A	I	A	I	A	I	A
Low	0.95	0.85	0.3	0.25	0.5	0.15	0.1	0.05
High	0.05	0.15	0.7	0.75	0.5	0.85	0.9	0.95

Table 6.14: Conditional Probability table for Drunkard node.

Drunkard								
20. Person is moving around in a drunken manner	Inactive				Active			
21.Person has a beer in his hand	Inactive		Active		Inactive		Active	
22.Person talks rubbish to others	I	A	I	A	I	A	I	A
Low	0.95	0.9	0.9	0.2	0.25	0.1	0.05	0.05
High	0.05	0.1	0.1	0.8	0.75	0.9	0.95	0.95

Table 6.15: Conditional Probability table for Theft node.

Theft								
13.People are in a quarrel	Inactive				Active			
19.Person accusing other of pickpocketing	Inactive		Active		Inactive		Active	
11.Person is talking loud	I	A	I	A	I	A	I	A
Low	0.95	0.75	0.35	0.3	0.3	0.15	.01	0.05
High	0.05	0.25	0.65	0.7	0.7	0.85	0.9	0.95

Table 6.16: Conditional Probability table for Vomiting node.

Vomiting								
5.Person is puking	Inactive				Active			
8.Person is lying on the walkpath	Inactive		Active		Inactive		Active	
2.Person is providing help	I	A	I	A	I	A	I	A
Low	0.95	0.7	0.85	0.8	0.2	0.2	0.1	0.05
High	0.05	0.3	0.15	0.2	0.8	0.8	0.9	0.95

Table 6.17: Conditional Probability table for Abandoned_Luggage node.

Abandoned_Luggage		
18.Luggage is lying around	Inactive	Active
Low	0.95	0.2
High	0.05	0.8

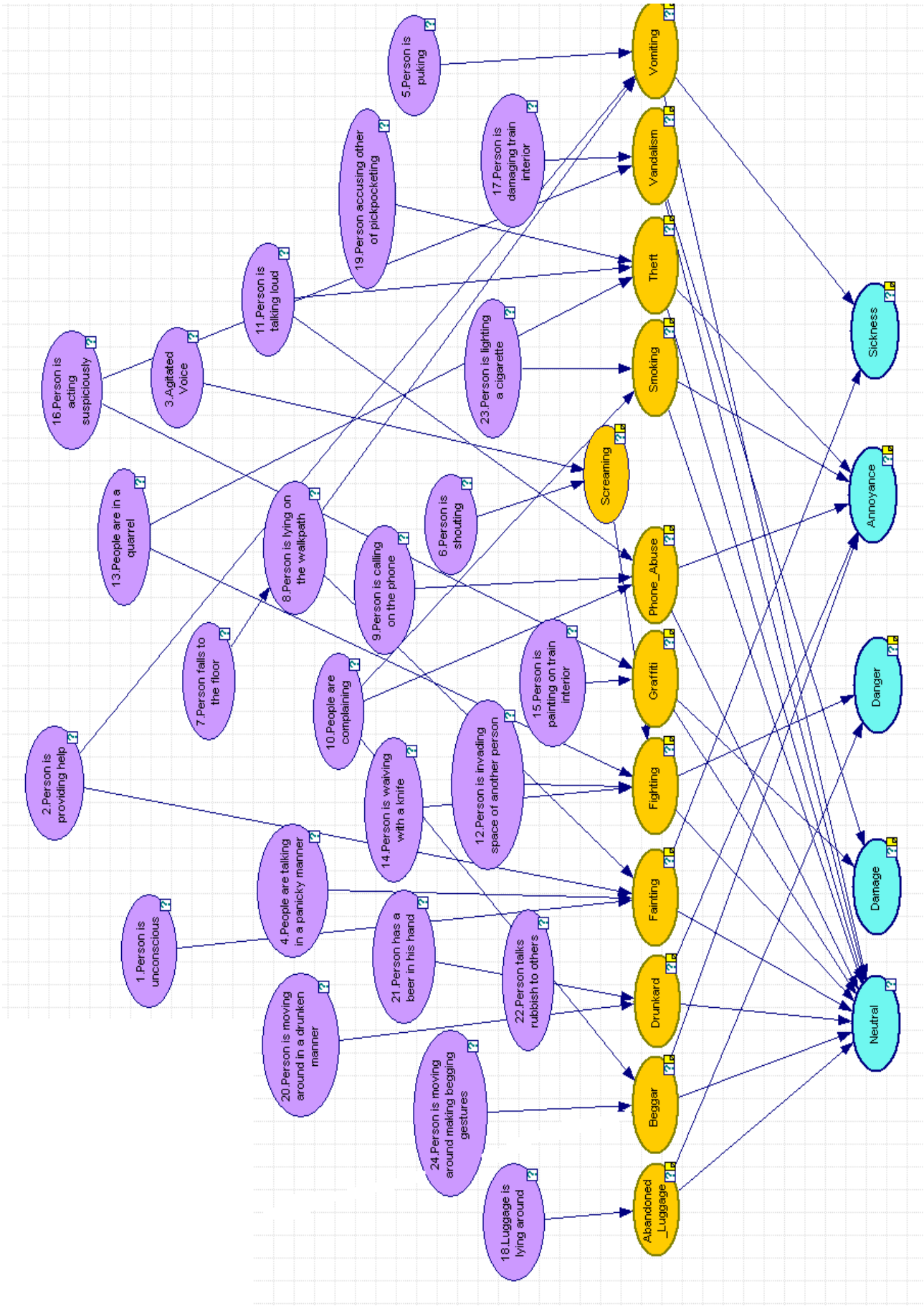


Figure 6.1: The Bayesian Network consists of five scenario output nodes, which receives its input from high-level feature- and situation nodes.

6.4 The hidden layer

For our Bayesian Network the input nodes are the high-level features we compiled in the first section. However above that layer of nodes we can have the hidden layer which will represent the low-level features of which these higher-level features are composed of. Figure 6.2 gives just an example of an extra layer. The nodes here represent a subnetwork itself. Figure 6.3 gives how the *sensormanagement* subnetwork could have been composed of. It would contain all the sensors available in such a train compartment, which would register all the low-level features. The train subnetwork could contain a node if the train is moving for example. The subnetwork *Schedule* could contain nodes for Time of the Day, Train is on time, Train arriving at station X. Figure 6.4 shows just a screenshot of a possible arrangement of the schedule subnetwork. More detail can be given for this hidden layer of features, but that is just beyond the scope of our research. But it is good to know that in fact it all starts with data we get from the sensors(audio/video), which can then be fused together to come to the lower- and higher-level features.

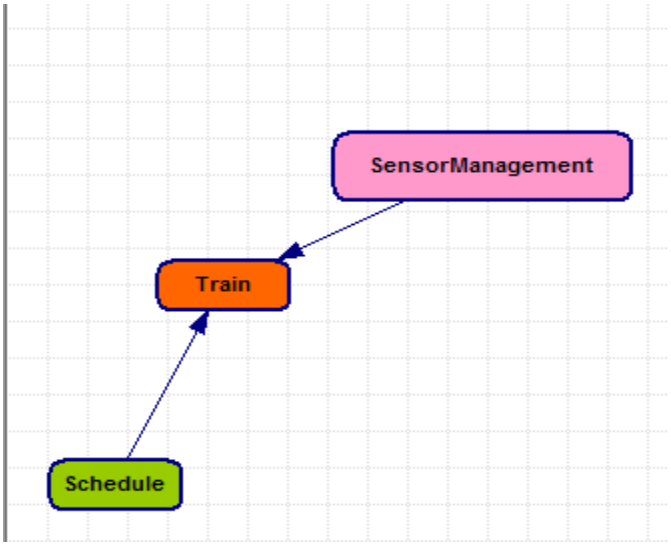


Figure 6.2: Example of a hidden layer.

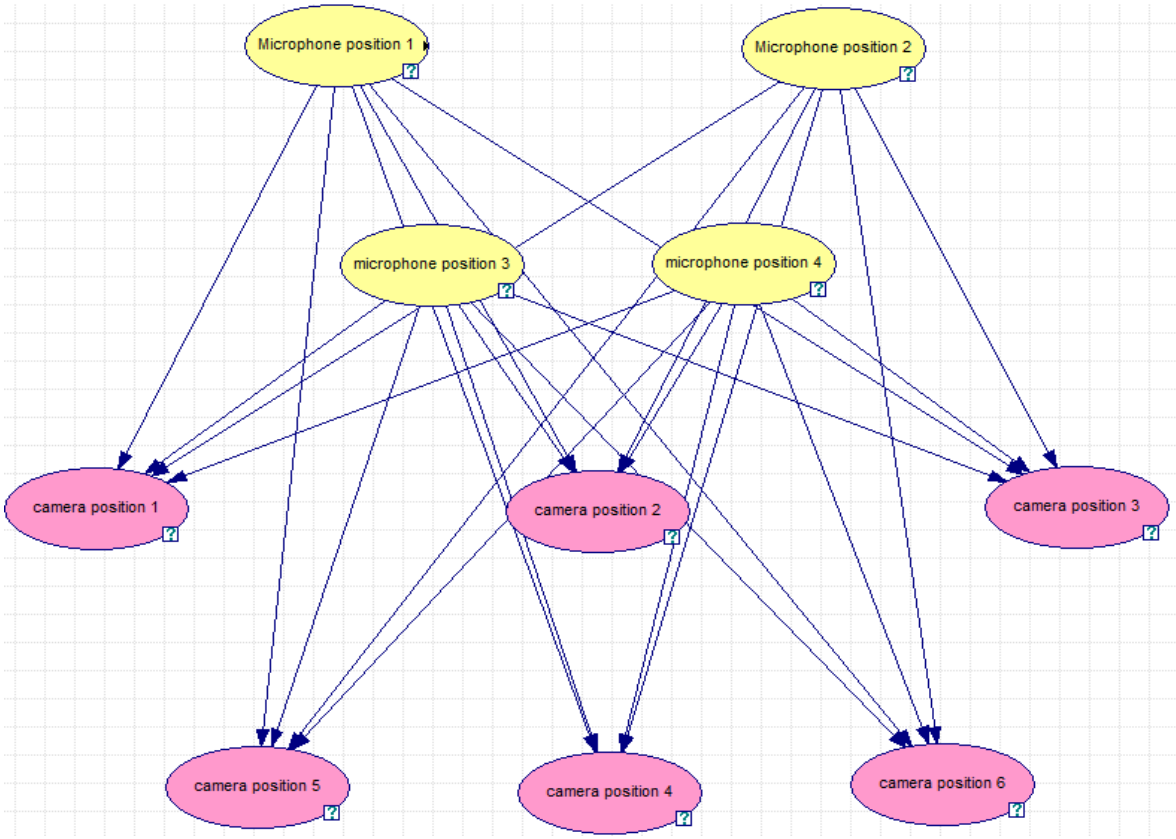


Figure 6.3: Sensor management nodes.

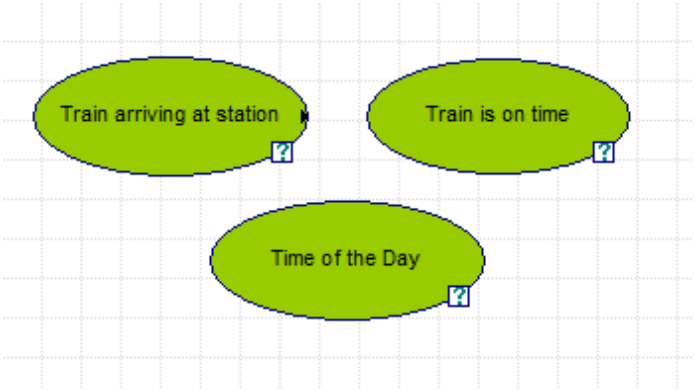


Figure 6.4: Example of possible nodes for the Schedule subnetwork.

6.5 Conclusion

The designed network will be used in a train compartment. During our research we came to the following setup:

- In every train compartment we have the designed bayesian network.
- The five scenario nodes are all connected to a central functional node.
- The functional node filters of all the incoming nodes for example to rank the highest 6 probabilities.
- These could then be showed on six monitors of a human operator, which can then if required take appropriate actions.

Figure 6.5 visualizes in simple view our setup. The functional node can have any function to filter the highest probability scenarios from all compartments incorporating this Bayesian Network. The sensors(camera/audio) of the filtered compartments by the functional node can then be showed on the screens of the human operators.

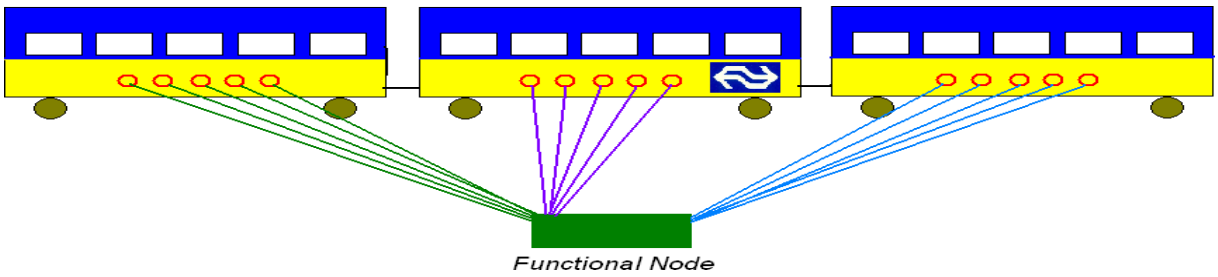


Figure 6.5: BN setup in train compartments.

This setup could be a probabilistic decision support system in aid of the human operators monitoring the situations in the train from a remote location. Probabilistic DSSs (Decision Support systems) are based on a philosophically different principle than rule-based expert systems. While the latter attempt to model the reasoning of a human expert, the former use an axiomatic theory to perform computation. The soundness of probability theory provides a clear advantage over rule-based systems that usually represent uncertainty in an ad-hoc manner, such as using certainty factors, leading to under-responsiveness or over-responsiveness to evidence and possibly incorrect conclusions.

We analyzed which high-features are essential for our situations and compiled a list which we then implemented in our bayesian network. We tried to limit the incoming arcs from parent nodes to at the most 4, to keep the conditional probability tables as small as possible. We used the data from the aggression survey held amongst frequent and daily commuters of trains as a initial for our

conditional probability tables where applicable. For other nodes we played the role of a human expert in the field to compute our initial CPTs.

In the next chapter we will do a case study, using some footage shot during a test with actors in a real train environment. The actors played several scenarios that can occur in a environment. We will base the case study on this material and will analyze some scenarios using our Bayesian Network.

Chapter 7

A case study in a real-life context using the Bayesian Network

7.1 Introduction

In this chapter we will discuss several cases, which are in this case scenarios, that can occur in real life. We have real-life footage of actual events(scenarios) which we will take as the basis for our study. We will extract and analyze the information and high-level features from the footage, insert them into our designed Bayesian Network and report and evaluate the results. Each of the following section we will present and discuss one case.

7.2 Case I: Medical Emergency

7.2.1 Analyzing the footage

The first case we will discuss involves a medical emergency situation occurring in the train. The footage for this scenario is approximately 2:17 minutes long. We present the important frames occurring in the footage where we will analyze and extract some features we recognize and add them into our Bayesian Network as evidence. We will then let the network infer using the Clustering inference algorithm, which is the best known exact algorithm, to calculate the probabilities for both the situation and the scenario layer of nodes. Depending on the features we activate as evidence we will evaluate the result of scenario probabilities and see if the proper scenario is the most likely. Let's start to analyze the different frames below.

The footage starts as depicted in figure 7.1, where it shows people sitting(green squares) and two persons walking (red square) in the walkpath, probably to find

Figure 7.1: Case 1: Two people walking in the walkpath.

a seat.

Figure 7.2: Case 1: Person falling to the floor.

In figure 7.2, a person suddenly falls to the floor (red square). Due to this sudden act people start to talk in a panicky manner (green squares). Some are even shouting and talking with a agitated voice (short time).

In figure 7.3, a person is providing help to the person lying on the floor (red square). Other person is still talking in a panicky manner (green square).

Figure 7.4 shows a person still providing help (red square) and a passenger is making a call on a mobile phone (green square) and talking loud, while another person is still acting in a panicky matter (blue square).

7.2.2 Activating the features

We will discuss, using the transcript from previous section, which of the features we could identify and we need to use (activate) as evidence for our Bayesian Network. Table 7.1 shows per frame presented in previous section which of the features are identified.

Figure 7.3: Case 1: Person is providing help.

Figure 7.4: Case 1: Person is calling on a mobile phone.

Table 7.1: Extracted high-level features from footage of Case 1

Figure	Feature
7.2	7.Person falls to the floor 8.Person is lying on the walkpath(Propagated evidence) 6. Person is shouting
7.3	2.Person is providing help. 4. People are talking in a panicky manner.
7.4	9. Person is calling on a phone. 11. Person is talking loud.

7.2.3 Evaluation

Using our initially calculated conditional probability tables we inserted the evidence found in this footage and let the Bayesian Network do its inference and we will evaluate the result of the scenario. Figure 7.5 shows the bar chart of all the nodes in the *situation* (yellow) and *scenario* (blue) layers. If you just

look at the end result(blue bars), e.g we want to know the scenario, we can immediately see that the sickness scenario has the highest probability given all the events that occurred in the train compartment. If we look at how this probability was inferred we look at the layer above (situation layer). If we want to know which situation is most likely to occur we look at their probabilities. We see three situations which have a high probability. These are *Screaming*, *Fainting* and *Phone_Abuse*. The reason that the situation Phone_Abuse has a reasonably high probability is the fact that in the footage there was a person calling on the phone and he was talking a bit loud. In the same way screaming had a high probability of occurring was the fact of emotionally voice and and bit of panic shouts.

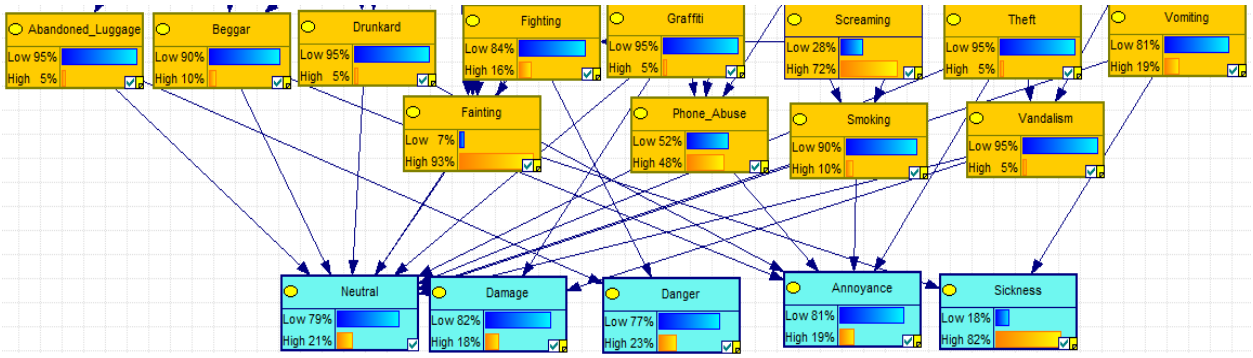


Figure 7.5: Case 1: Resulting probabilities after inference.

Although there were more situations with high probabilities, the Bayesian Network was still able to correctly infer that fainting and consequently the sickness scenario are most likely to be the case.

7.3 Case II: Mobile Phone

7.3.1 Analyzing the footage

The second case we will discuss covers the mobile phone abuse situation which is categorized into the annoyance scenario. The footage for this scenario is approximately 1:34 second long. We will again go through several frames as a example of features we recognize and then later activate them in our Bayesian Network.

The footage starts as depicted in figure 7.6, where people are sitting quietly doing their things, while the mobile phone of a person starts to ring (red square). The person is talking very loud as if he is alone somewhere.

In figure 7.7 after a couple of seconds of loud talking, other people (red squares) start complain and ask if he (green square) could keep it down. After a long debate he agrees and hangs up the phone without further discussion.

Figure 7.6: Case 2: Person talking on the phone.

Figure 7.7: Case 2: People complaining to the person calling.

This scenario is fairly short as it was solved without any complications further more. There could have been a quarrel of stubbornness of the caller ignoring all complaints, which could have made the scenario much more complicated. But for now this is the scenario as it was played and recorded. In the next section we will give the features for this case.

7.3.2 Activating the features

We present, using the transcript from previous section, which of the features we could identify and we need to use (activate) as evidence for our Bayesian Network. Table 7.2 show per frame presented in previous section which of the features are identified.

7.3.3 Evaluation

If we looking at the end result(blue bars) in figure 7.8, e.g we want to know the scenario, we can immediately see that the annoyance scenario has the highest

Table 7.2: Extracted high-level features from footage of Case 2

Figure	Feature
7.6	9. Person is calling on a phone. 11. Person is talking loud.
7.7	9. Person is calling on a phone. 10. People are complaining.

probability given all the events that occurred in the train compartment. However the probability of occurring doesn't look that high with only a chance of 0.46. But relative to the other scenario nodes it is the highest. The reason we think that this chance doesn't seem too confident is because of the parent nodes of the annoyance node. Given the input features only one of the parent nodes is very confident with a high probability. This is the phone_Abuse node with a probability of 0.91. So given the Bayes rule of conditional probability results in a 0.46 probability of the scenario being an annoyance.

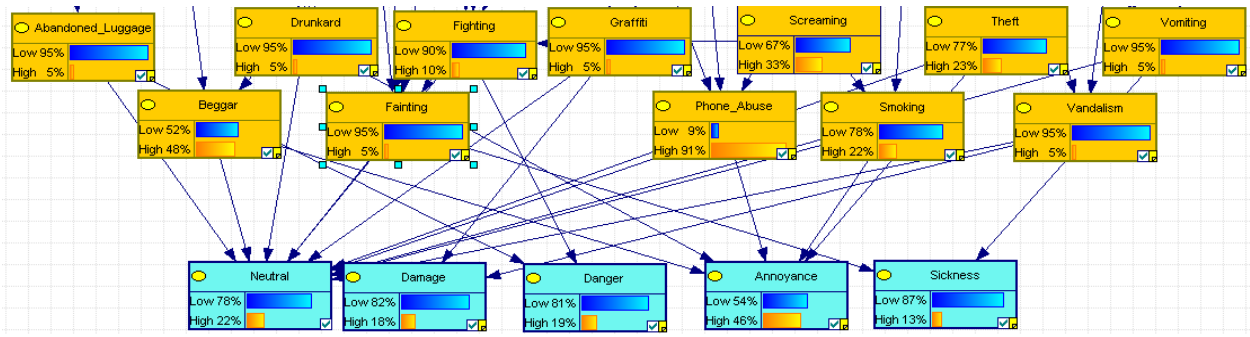


Figure 7.8: Case 2: Resulting probabilities after inference.

If the conditional probability tables would have been learned with more empirical data, the results would be more accurate of course, however given the current situation, the outcome was indeed satisfying, because it predicted to actual scenario quiet good.

7.4 Case III: Beggar

7.4.1 Analyzing the footage

The third and last case we will discuss in this thesis covers the begging situation which is categorized in the annoyance category. The footage for this scenario is approximately 1:17 second long. We will again go through several frames as a example of features we recognize, annotate them accordingly and then later add them as evidence in our Bayesian Network.

The footage starts with a person (red square) walking in the walkpath, stopping

temporarily on this way stretching his hands and asking for something to people sitting (green squares)(Figure 7.9)

Figure 7.9: Case 3: Person making begging gestures.

the footage continues as depicted in figure 7.10, where people are complaining (red square) about his begging behavior (green square) and asking him to go work for his money.

Figure 7.10: Case 3: People are complaining about his behaviour.

In figure 7.11 the person making the begging gestures invades the personal space of another person (red square), which results in a quarrel between the two persons (green square). They are talking loud and even shouting. After another person (purple square) come between the two, the beggar moves on and keeps on making begging gestures along his way.

7.4.2 Activating the features

We present, using the transcript from previous section, which of the features we could identify and we need to use (activate) as evidence for our Bayesian

Figure 7.11: Case 3: Person is invading personal space.

Network. Table 7.3 shows per frame presented in previous section which of the features are identified.

Table 7.3: Extracted high-level features from footage of Case 3

Figure	Feature
7.9	24. Person is moving around making begging gestures.
7.10	10. People are complaining.
7.11	12. Person is invading space of another person. 11. Person is talking loud. 6. Person is shouting. 13. People are in a quarrel.

7.4.3 Evaluation

Let us evaluate the output of all the evidence we inserted in our Bayesian Network. Figure 7.12 shows the result of the scenario (blue) and situation (yellow) layers. The situation nodes with the highest probabilities are *Beggar* and *Fighting* with respectively 0.95 and 0.57. The reason for the begging situation is obvious given the input feature nodes we annotated, but the fighting situation has such a high probability due to the fact that for a period of time there was a quarrel between the beggar and a passenger and they were shoving each other a little bit. Along with invasion of personal space and loud talking, the Bayesian Network picked that up and inferred a probability of 0.57. The structure of the network is such that some of the features annotated here have an influence on the fighting situation. Nevertheless the network did not infer the situation to be a high fighting situation, but inferred it to about a 50-50% chance of the situation being a fighting one.

Due to the fact that both beggar and fighting situation nodes have a high probability, but also the beggar situation probability was higher then that of the fighting situation, that didn't reflect quiet well into the scenario nodes(output). There we see that the Danger scenario was more likely that the annoyance scenario, meaning fighting is had a bigger influence then the beggar situation.

There could have been couple of reasons that this happened. First, the CPT's weren't that well adjust for the two scenario nodes, but we checked here and the CPT's reflect that if a dangerous situation occurs, that it should be awarded a priority over the other scenarios. This could be a nice feature, as if such a combination of more scenarios or situations happens in a train environment, we want to know the ones where the probability of the aggression level is high and to have a human operator re-analyze the situation and act accordingly. Another reason could be the same we discussed with case 2. The number of nodes that influence the Annoyance node is higher then that of the Danger node.

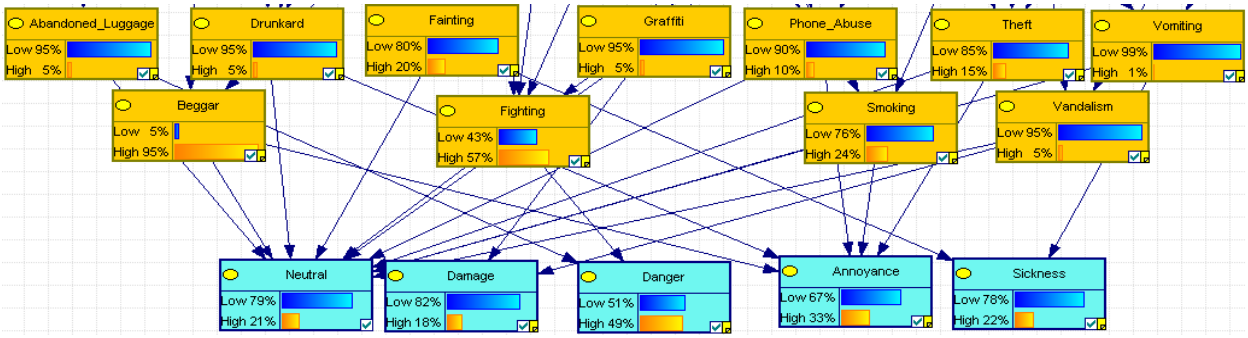


Figure 7.12: Case 3: Resulting probabilities after inference.

7.5 Conclusion

In this chapter we presented 3 cases, which we analyzed and then annotated. In turn we inserted the evidence into our designed Bayesian Network, which inferred the output probabilities of both the situation and scenario nodes. Although we managed to compute and analyze all the conditional probability tables by hand, which was a tedious job, we can be quiet happy with the results the network is spitting out.

If there was a large amount of empirical data available, we could use that to learn the conditional probability tables using one of the many known learning algorithms. This would improve the accuracy of the inference. In the networks current state and shown using the 3 cases presented in this case, it can be shown that the job it was designed to do can be done in a satisfying way.

Chapter 8

Conclusion and Future work

8.1 Conclusion

The **goals** of this thesis were as stated in section 1.3: (a) the development of an aggression ontology which models the main concepts in a train compartment and the interrelations between the concepts, (b) the design and development of a Bayesian Network(BN) which will incorporate the different uncertainty levels of the features, situation and ultimately the scenario in the train.

To meet these goals, the following **research objectives** were established and achieved:

1. Gather knowledge of the Dutch Railways current aggression definition and categorization. Also we researched how the literature approached human aggression. In chapter 2 we cover the different point of views of human aggression. On one side we saw it from a psychological point of view and also from a biological point of view. And try to formulate a good definition for human aggression in general.

2. The categorization of the different forms of aggression known the in research field of aggression. In section 2.3 we presented 3 types of known aggression that are extensively being and have been researched by both psychological and biological groups. These are the *Instrumental-, Affective- and Reactive* aggression.

3. The meaning and understanding of ontologies as well as the range of purposes that ontologies may serve (chapter 3). Research the best way ontologies can be built and the appropriate software to use for the use of our Aggression ontology. Compare the different ontology languages used for designing an ontology in. Ontologies can be distinguished into several types of ontologies and we researched what the purpose of our goal is in this thesis and chose the appropriate type.

4. The representation of our data gathered from footage shot in a real train environment and implement this into a Bayesian Network. In order to achieve this, chapter 5 presents background information gathered in research about Bayes' rule and probability. The representation of network structure and conditional- and joint probability is covered(section 5.1.2, along with a small overview of how inference can be done using several algorithms. There are also different modeling software available in the aid to design a BN and infer with it. A small comparison between those was discussed.

The **specific tasks**, undertaken to meet the objectives, and the results obtained are:

I. A literature research study [20] was carried out to approach the nature of human aggression in general as well as viewed by different other research fields. We came to a distinction of aggression into several types(Ch 3). The conclusion was that aggressive behavior, is shaped and developed through learning processes. Both genetic inheritance and learned tendencies serve as predisposing background conditions for aggression, which is a response to provoking conditions in the persons environment.

II. The aggression ontology was analyzed and implemented using the Protege ontology editor. This ontology incorporates the main concepts of a train compartment, which is the domain of the ontology and also domain of this thesis. The ontology can be seen as a general ontology with high-level concepts as those are the core features of this thesis. The main classes and sub-classes of the ontology were designed. Then we had the object-properties relate the several classes which each other in such a way that we semantically built our train domain.

III. A Bayesian Network is designed and applied to the train domain as a support decision system, to aid in the analysis of aggression in the train. The Bayesian Network takes high-level features as input and when activated, the corresponding situations and consequently a scenario probability will be obtained through the conditional probability tables.

IV. A case study using footage of live scenarios has been carried out using the designed Bayesian Network (ch.7). Real empirical data was not available, instead we analyzed some scenario footage with frame by frame and took out the relevant high-level features and feed those as a input to our Bayesian Network and evaluated the result.

8.2 Recommendations for future work

We would like to conclude this thesis with some open research lines to be considered for future work, based on the experience from our research. These are listed below:

- Construction of a sub-ontology listing more details and have the train compartment concepts in more detail(for example, windows, toilets, exit door, emergency exits etc.) Other details which are important for the lower-level features as described in section 6.4 could be designed in a separate sub-ontology too. Ontologies can easily coupled together, using namespaces. Our designed aggression ontology can be used as a good basis for extension.

- transform the aggression ontology to be able to incorporate probabilities in OWL. There are currently different research going on using probabilistic extensions to the ontology language OWL, like PR-OWL [12] and OntoBayes [15].

- Use the ontology as a reasoning engine rather then using a Bayesian Network. There are several methods using rule based approaches. This might not be as affective as expert system approach but more insight in the capabilities of a reasoning ontology would not hurt. A combination of ontology reasoning and a expert system like a Bayesian Network could be preferred also.

- Transform the Bayesian Network designed in chapter 6 into a Dynamic Bayesian Network(DBN). This would have the advantage of temporal reasoning enabled and a situation or scenario can be monitored over a period of time and see how it plays out. A simple example; if we look back at our case study II: phone abuse, where the situation got resolved easily after a period in time. If there was a DBN involved with reasoning in time, the feature that were activated at t=5, would have been disabled or lowered in probability at t=n, and the scenario would have been inferred to be Neutral again.

- Gather a large amount of empirical data and use that to learn the conditional probability tables of the nodes in the situation and scenario layers. This would improve the accuracy of the output in a positive way.

Bibliography

[1] Nederlandse Spoorwegen Jaarrapport 2006. http://www.ns.nl/www.ns.nl/pdf/ns_jvs_06_web.pdf.

[2] L. Berkowitz. *Aggression: Social Psychological Analysis, A*. McGraw-Hill: New York, 1962.

[3] L. Berkowitz. *Aggression: Its Causes, Consequences and Control*. NY:Mcgraw-Hill, 1993.

[4] J.A Bilmes and G.G. Zweig. The graphical models toolkit: An opensource software system for speech and time-series processing. In *2002 IEEE international conference on acoustics, speech, and signal processing (ICASSP02)*, 2002.

[5] K. Bjorkqvist, K.M.J. Lagerspetz, and A. Kaukiainen. Do girls manipulate and boys fight? *Aggressive Behavior*, 18(117-127), 1992.

[6] C. M. Borduin. Multisystemic treatment of criminality and violence in adolescents. *Journal of the American Academy of Child & Adolescent Psychiatry*, 38(3):250–255, march 1999.

[7] W. Borst. Construction of engineering ontologies. *PhD Thesis, University of Twente*, 1997.

[8] A.H. Buss. *The psychology of aggression*. Wiley, 1961.

[9] G.V. Capara, C Barbaranelli, and P.G Zimbardo. Understanding the complexity of human aggression: Affective, cognitive, and social dimensions of individual differences in propensity towards aggression. *European Journal of Personality*, 10:133–155, 1996.

[10] J. Chen and M.J. Drudzel. Bn-ais: An adaptive importance sampling algorithm for evidential reasoning in large bayesian networks. *Journal of Arificial Intelligence Search*, 13:155–188, 2000.

[11] World Wide Web Consortium. *http://www.w3.org/*.

[12] P.C.G. Costa and K.B. Laskey. Pr-owl: A framework for probabilistic ontologies. *in Proceedings of the 4th International conference on formal ontology in Information Systems*, 2006.

[13] F. Cupillard, A. Avanzi, and F. Brémond. Video understanding for metro surveillance. *in Proceedings of the IEEE International Conference on Networking, Sensing and Control*, 2004.

[14] I. Horrocks, D. Connolly, F. van Harmelen, D.L. McGuiness, P.F Patel-Schneider, and L.A. Stein. Daml + oil (march 2001) reference description. *http://www.w3.org/TR/daml+oil-reference*, 2001.

[15] Z. Ding and Y. Peng. A probabilistic extension to ontology language owl. *in Proceedings of the 37th Hawaii International Conference on System Sciences*, 2004.

[16] K.A. Dodge and J.D. Coie. Social information processing factors in reactive and proactive aggression in children's peer groups. *Journal of Personality and Social Psychology*, 53:1145–1158, 1987.

[17] The Protégé Ontology Editor and Knowledge Acquisition System. http://protege.stanford.edu/.

[18] S. Feshbach. The function of aggression and the regulation of aggressive drive. *Psychological review*, 71:257–272, 1964.

[19] A. Ganga. A world model for aggression detection in trains using a multi-modal camera. *Technical report, MMI Group, Delft University of Technology*, 2006.

[20] V. Gangaram Panday. Multi-modal surveillance in public places. *Technical report, MMI Group, Delft University of Technology*, 2007.

[21] K.A. Gleason, L.A. Jensen-Campbell, and D.S. Richardson. Agreeableness as a predictor of aggression in adolescence. *Wiley InterScience*, 30:43 – 61, 2004.

[22] T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.

[23] D.E. Heckerman. A tutorial on learning with bayesian networks. *in Proceedings of the NATO advanced study institute on learning in graphical models*, 1998.

[24] Uniform Resource Identifiers. *http://en.wikipedia.org/wiki/Uniform_Resource_Identifier*.

[25] A. Kivela and E. Hyvonen. Ontological theories for the semantic web. *HIIT Publications*, 2002.

[26] C. Lacave and F.J. Diez. A review of explanation of methods for bayesian networks. *Knowlegde Engineering Review*, 17, 2002.

[27] S.K. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their applicaton to expert systems. *Journal of Royal statistical Society, Series B (Methodological)*, 50(2):157–244, 1988.

[28] B.P. Lo, J. Sun, and S.A. Velastin. Fusing visual and audio information in a distributed intelligent surveillance system for public transport systems. *ACTA Automatica Sinica*, 29(3):393–407, 2003.

[29] C.J. Mathaus, M.M Kokar, and K. Baclawski. A core ontology for situation awareness. *Sixth Internation conference on Information Fusion*, July 2003.

[30] C. Matuszek, J. Cabral, M. Witbrock, and J. DeOliveira. An introduction to the syntax and content of cyc. *in Proceedings of the 2006 AAAI spring symposium on Formalizing an compiling Background knowledge and its applications to knowledge representation and question answering*, 2006.

[31] K.A. Miczek, E. Weerts, M. Haney, and J. Tidey. Neurobiological mechanisms controlling aggression: preclinical developments for pharmacotherapeutic interventions. *Neuroscience and biobehavioral reviews*, 18(1):97–110, May 1994.

[32] M. Ismail. Video content analysis and aggression detection system for a train environment. Master's thesis, MMI Group, Delft University of Technology, 2007.

[33] K.P. Murphy. The bayes net toolbox for matlab. *Computing science and statistics*, 33, october 2001.

[34] L.G. Ost, A. Jerremalm, and J. Johansson. Individual response patterns and the effects of different behavioral methods in the treatment of social phobia. *Behavior research and therapy*, 19(1):1–16, 1981.

[35] V. Pavlovic, J.M. Rehg, and T. Cham. A dynamic bayesian network approach to figure tracking using learned dynamic models. *in Proceedings of the International conference on computer vision*, 1999.

[36] J. Pearl. Fusion, propagation, and stucturing in belief networks. *Artifial Intelligence*, 1986.

[37] IST Project. http://wonderweb.semanticweb.org/.

[38] J. Pulkinnen. Proactive and reactive aggression in early adolescence as precursors to anti- and prosocial behavior in young adults. *Aggressive Behavior*, 22:241–57, 1996.

[39] RacerPro Reasoner. http://www.racer-systems.com/products/download/index.phtml.

[40] P. Sember and I. Zukerman. Strategies for generation micro explanation for bayesian belief networks. *Uncertainty in Artificial Intellegence*, 5, 1990.

[41] A. Simon. Aggression in a prison setting as a function of lunar phases. *Psychological Reports*, 82(3):747–752, 199.

[42] M.K. Smith, C. Welty, and D.L. McGuinness. *OWL Web Ontology Language Guide, W3C.* http://www.w3.org/2004/OWL/, 2004.

[43] J. Sonnemans and N.H. Frijda. The structure of subjective emotional intensity. *Cognition and Emotion*, 8(4):329–350, 1994.

[44] A.N. Steinberg and D.B. Jacobs. Rethinking the jdl data fusion levels. *in Proceedings of the National Symposium on Sensor and data Fusion*, 2004.

[45] R. Studer, V. Benjamins, and D. Fensel. Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, 1998.

[46] WonderWeb OWL Ontology Validator. http://www.mygrid.org.uk/owl/validator.

Appendix A

Complete OWL Code of the Aggression Ontology

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<rdf:RDF xmlns="http://www.owl-ontologies.com/OntologyFinal.owl#"
     xml:base="http://www.owl-ontologies.com/OntologyFinal.owl"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#">
    <owl:Ontology rdf:about=""/>
    <owl:Class rdf:ID="Abandoned_Luggage">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasObject"/>
                <owl:someValuesFrom rdf:resource="#Luggage"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf rdf:resource="#Situation"/>
        <owl:disjointWith rdf:resource="#Beggar"/>
        <owl:disjointWith rdf:resource="#Vomiting"/>
```

```
        <owl:disjointWith rdf:resource="#Phone_Abuse"/>
        <owl:disjointWith rdf:resource="#Vandalism"/>
        <owl:disjointWith rdf:resource="#Smoking"/>
        <owl:disjointWith rdf:resource="#Drunkard"/>
        <owl:disjointWith rdf:resource="#Fighting"/>
        <owl:disjointWith rdf:resource="#Fainting"/>
        <owl:disjointWith rdf:resource="#Screaming"/>
        <owl:disjointWith rdf:resource="#Theft"/>
        <owl:disjointWith rdf:resource="#Graffiti"/>
    </owl:Class>
    <owl:Class rdf:ID="Aggressor">
        <rdfs:subClassOf rdf:resource="#Passenger"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasObject"/>
                <owl:someValuesFrom rdf:resource="#HumanObject"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Annoyance">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasSituation"/>
                <owl:someValuesFrom>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                            <owl:Class rdf:about="#Beggar"/>
                            <owl:Class rdf:about="#Drunkard"/>
                            <owl:Class rdf:about="#Phone_Abuse"/>
                            <owl:Class rdf:about="#Smoking"/>
                            <owl:Class rdf:about="#Theft"/>
                        </owl:unionOf>
                    </owl:Class>
                </owl:someValuesFrom>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf rdf:resource="#Scenario"/>
        <owl:disjointWith rdf:resource="#Danger"/>
        <owl:disjointWith rdf:resource="#Neutral"/>
        <owl:disjointWith rdf:resource="#Sickness"/>
        <owl:disjointWith rdf:resource="#Damage"/>
    </owl:Class>
    <owl:Class rdf:ID="Beggar">
```

```
        <rdfs:subClassOf rdf:resource="#Situation"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#involvedInSituation"/>
                <owl:someValuesFrom rdf:resource="#Aggressor"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <owl:disjointWith rdf:resource="#Graffiti"/>
        <owl:disjointWith rdf:resource="#Abandoned_Luggage"/>
        <owl:disjointWith rdf:resource="#Vandalism"/>
        <owl:disjointWith rdf:resource="#Phone_Abuse"/>
        <owl:disjointWith rdf:resource="#Fighting"/>
        <owl:disjointWith rdf:resource="#Vomiting"/>
        <owl:disjointWith rdf:resource="#Screaming"/>
        <owl:disjointWith rdf:resource="#Drunkard"/>
        <owl:disjointWith rdf:resource="#Theft"/>
        <owl:disjointWith rdf:resource="#Smoking"/>
        <owl:disjointWith rdf:resource="#Fainting"/>
    </owl:Class>
    <owl:Class rdf:ID="Book">
        <rdfs:subClassOf rdf:resource="#HumanObject"/>
        <owl:disjointWith rdf:resource="#Toy"/>
        <owl:disjointWith rdf:resource="#Ticket"/>
        <owl:disjointWith rdf:resource="#Cigarette"/>
        <owl:disjointWith rdf:resource="#Paint"/>
        <owl:disjointWith rdf:resource="#Luggage"/>
        <owl:disjointWith rdf:resource="#Bottle"/>
        <owl:disjointWith rdf:resource="#Knife"/>
        <owl:disjointWith rdf:resource="#Phone"/>
    </owl:Class>
    <owl:Class rdf:ID="Bottle">
        <rdfs:subClassOf rdf:resource="#HumanObject"/>
        <owl:disjointWith rdf:resource="#Cigarette"/>
        <owl:disjointWith rdf:resource="#Luggage"/>
        <owl:disjointWith rdf:resource="#Book"/>
        <owl:disjointWith rdf:resource="#Phone"/>
        <owl:disjointWith rdf:resource="#Ticket"/>
        <owl:disjointWith rdf:resource="#Toy"/>
        <owl:disjointWith rdf:resource="#Knife"/>
        <owl:disjointWith rdf:resource="#Paint"/>
    </owl:Class>
    <owl:Class rdf:ID="Cigarette">
        <rdfs:subClassOf rdf:resource="#HumanObject"/>
```

```
        <owl:disjointWith rdf:resource="#Toy"/>
        <owl:disjointWith rdf:resource="#Bottle"/>
        <owl:disjointWith rdf:resource="#Paint"/>
        <owl:disjointWith rdf:resource="#Phone"/>
        <owl:disjointWith rdf:resource="#Ticket"/>
        <owl:disjointWith rdf:resource="#Book"/>
        <owl:disjointWith rdf:resource="#Luggage"/>
        <owl:disjointWith rdf:resource="#Knife"/>
    </owl:Class>
    <owl:Class rdf:ID="Conductor">
        <rdfs:subClassOf rdf:resource="#Person"/>
    </owl:Class>
    <owl:Class rdf:ID="Damage">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasSituation"/>
                <owl:someValuesFrom>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                            <owl:Class rdf:about="#Graffiti"/>
                            <owl:Class rdf:about="#Vandalism"/>
                        </owl:unionOf>
                    </owl:Class>
                </owl:someValuesFrom>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf rdf:resource="#Scenario"/>
        <owl:disjointWith rdf:resource="#Annoyance"/>
        <owl:disjointWith rdf:resource="#Danger"/>
        <owl:disjointWith rdf:resource="#Neutral"/>
        <owl:disjointWith rdf:resource="#Sickness"/>
    </owl:Class>
    <owl:Class rdf:ID="Danger">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasSituation"/>
                <owl:someValuesFrom>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                            <owl:Class rdf:about="#Abandoned_Luggage"/>
                            <owl:Class rdf:about="#Fighting"/>
                            <owl:Class rdf:about="#Screaming"/>
                        </owl:unionOf>
```

```
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#Scenario"/>
    <owl:disjointWith rdf:resource="#Annoyance"/>
    <owl:disjointWith rdf:resource="#Neutral"/>
    <owl:disjointWith rdf:resource="#Sickness"/>
    <owl:disjointWith rdf:resource="#Damage"/>
</owl:Class>
<owl:Class rdf:ID="DomainConcept"/>
<owl:Class rdf:ID="Drunkard">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#involvedInSituation"/>
            <owl:someValuesFrom rdf:resource="#Aggressor"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#Situation"/>
    <owl:disjointWith rdf:resource="#Abandoned_Luggage"/>
    <owl:disjointWith rdf:resource="#Smoking"/>
    <owl:disjointWith rdf:resource="#Fainting"/>
    <owl:disjointWith rdf:resource="#Fighting"/>
    <owl:disjointWith rdf:resource="#Theft"/>
    <owl:disjointWith rdf:resource="#Graffiti"/>
    <owl:disjointWith rdf:resource="#Vomiting"/>
    <owl:disjointWith rdf:resource="#Vandalism"/>
    <owl:disjointWith rdf:resource="#Screaming"/>
    <owl:disjointWith rdf:resource="#Phone_Abuse"/>
    <owl:disjointWith rdf:resource="#Beggar"/>
</owl:Class>
<owl:Class rdf:ID="Fainting">
    <rdfs:subClassOf rdf:resource="#Situation"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#involvedInSituation"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <owl:Class rdf:about="#Aggressor"/>
                        <owl:Class rdf:about="#Passenger"/>
                    </owl:unionOf>
                </owl:Class>
```

```
                </owl:someValuesFrom>
            </owl:Restriction>
        </rdfs:subClassOf>
        <owl:disjointWith rdf:resource="#Screaming"/>
        <owl:disjointWith rdf:resource="#Beggar"/>
        <owl:disjointWith rdf:resource="#Phone_Abuse"/>
        <owl:disjointWith rdf:resource="#Fighting"/>
        <owl:disjointWith rdf:resource="#Vandalism"/>
        <owl:disjointWith rdf:resource="#Vomiting"/>
        <owl:disjointWith rdf:resource="#Graffiti"/>
        <owl:disjointWith rdf:resource="#Theft"/>
        <owl:disjointWith rdf:resource="#Abandoned_Luggage"/>
        <owl:disjointWith rdf:resource="#Drunkard"/>
        <owl:disjointWith rdf:resource="#Smoking"/>
    </owl:Class>
    <owl:Class rdf:ID="Fighting">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#involvedInSituation"/>
                <owl:someValuesFrom>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                            <owl:Class rdf:about="#Aggressor"/>
                            <owl:Class rdf:about="#Bottle"/>
                            <owl:Class rdf:about="#Conductor"/>
                            <owl:Class rdf:about="#Knife"/>
                            <owl:Class rdf:about="#Passenger"/>
                        </owl:unionOf>
                    </owl:Class>
                </owl:someValuesFrom>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf rdf:resource="#Situation"/>
        <owl:disjointWith rdf:resource="#Graffiti"/>
        <owl:disjointWith rdf:resource="#Vandalism"/>
        <owl:disjointWith rdf:resource="#Beggar"/>
        <owl:disjointWith rdf:resource="#Screaming"/>
        <owl:disjointWith rdf:resource="#Smoking"/>
        <owl:disjointWith rdf:resource="#Vomiting"/>
        <owl:disjointWith rdf:resource="#Drunkard"/>
        <owl:disjointWith rdf:resource="#Phone_Abuse"/>
        <owl:disjointWith rdf:resource="#Fainting"/>
        <owl:disjointWith rdf:resource="#Abandoned_Luggage"/>
```

```
        <owl:disjointWith rdf:resource="#Theft"/>
    </owl:Class>
    <owl:Class rdf:ID="Graffiti">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasObject"/>
                <owl:someValuesFrom rdf:resource="#Paint"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#involvedInSituation"/>
                <owl:someValuesFrom>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                            <owl:Class rdf:about="#Aggressor"/>
                            <owl:Class rdf:about="#TrainObject"/>
                        </owl:unionOf>
                    </owl:Class>
                </owl:someValuesFrom>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf rdf:resource="#Situation"/>
        <owl:disjointWith rdf:resource="#Screaming"/>
        <owl:disjointWith rdf:resource="#Beggar"/>
        <owl:disjointWith rdf:resource="#Theft"/>
        <owl:disjointWith rdf:resource="#Drunkard"/>
        <owl:disjointWith rdf:resource="#Fighting"/>
        <owl:disjointWith rdf:resource="#Vandalism"/>
        <owl:disjointWith rdf:resource="#Fainting"/>
        <owl:disjointWith rdf:resource="#Abandoned_Luggage"/>
        <owl:disjointWith rdf:resource="#Phone_Abuse"/>
        <owl:disjointWith rdf:resource="#Smoking"/>
        <owl:disjointWith rdf:resource="#Vomiting"/>
    </owl:Class>
    <owl:ObjectProperty rdf:ID="hasObject">
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <owl:Class rdf:about="#Person"/>
                    <owl:Class rdf:about="#Situation"/>
                </owl:unionOf>
            </owl:Class>
```

```
        </rdfs:domain>
        <rdfs:range rdf:resource="#HumanObject"/>
        <owl:inverseOf rdf:resource="#isObjectOf"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="hasSituation">
        <rdfs:domain rdf:resource="#Scenario"/>
        <rdfs:range rdf:resource="#Situation"/>
        <owl:inverseOf rdf:resource="#indicatesAScenario"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="HumanObject">
        <rdfs:subClassOf rdf:resource="#DomainConcept"/>
    </owl:Class>
    <owl:ObjectProperty rdf:ID="indicatesAScenario">
        <rdfs:domain rdf:resource="#Situation"/>
        <rdfs:range rdf:resource="#Scenario"/>
        <owl:inverseOf rdf:resource="#hasSituation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="involvedInSituation">
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <owl:Class rdf:about="#HumanObject"/>
                    <owl:Class rdf:about="#Person"/>
                    <owl:Class rdf:about="#TrainCompartment"/>
                    <owl:Class rdf:about="#TrainObject"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
        <rdfs:range>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <owl:Class rdf:about="#HumanObject"/>
                    <owl:Class rdf:about="#Person"/>
                    <owl:Class rdf:about="#Scenario"/>
                    <owl:Class rdf:about="#TrainObject"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:range>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="isObjectOf">
        <rdfs:domain rdf:resource="#HumanObject"/>
        <rdfs:range>
            <owl:Class>
```

```
            <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Person"/>
                <owl:Class rdf:about="#Situation"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:range>
    <owl:inverseOf rdf:resource="#hasObject"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isTravelingIn">
    <rdfs:domain rdf:resource="#Person"/>
    <rdfs:range rdf:resource="#TrainCompartment"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Knife">
    <rdfs:subClassOf rdf:resource="#HumanObject"/>
    <owl:disjointWith rdf:resource="#Phone"/>
    <owl:disjointWith rdf:resource="#Toy"/>
    <owl:disjointWith rdf:resource="#Ticket"/>
    <owl:disjointWith rdf:resource="#Cigarette"/>
    <owl:disjointWith rdf:resource="#Bottle"/>
    <owl:disjointWith rdf:resource="#Book"/>
    <owl:disjointWith rdf:resource="#Paint"/>
    <owl:disjointWith rdf:resource="#Luggage"/>
</owl:Class>
<owl:Class rdf:ID="Luggage">
    <rdfs:subClassOf rdf:resource="#HumanObject"/>
    <owl:disjointWith rdf:resource="#Phone"/>
    <owl:disjointWith rdf:resource="#Cigarette"/>
    <owl:disjointWith rdf:resource="#Knife"/>
    <owl:disjointWith rdf:resource="#Ticket"/>
    <owl:disjointWith rdf:resource="#Toy"/>
    <owl:disjointWith rdf:resource="#Book"/>
    <owl:disjointWith rdf:resource="#Bottle"/>
    <owl:disjointWith rdf:resource="#Paint"/>
</owl:Class>
<owl:Class rdf:ID="Neutral">
    <rdfs:subClassOf rdf:resource="#Scenario"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasSituation"/>
            <owl:cardinality rdf:datatype="&xsd;int">0</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#Annoyance"/>
```

```
        <owl:disjointWith rdf:resource="#Danger"/>
        <owl:disjointWith rdf:resource="#Sickness"/>
        <owl:disjointWith rdf:resource="#Damage"/>
    </owl:Class>
    <owl:Class rdf:ID="Paint">
        <rdfs:subClassOf rdf:resource="#HumanObject"/>
        <owl:disjointWith rdf:resource="#Cigarette"/>
        <owl:disjointWith rdf:resource="#Toy"/>
        <owl:disjointWith rdf:resource="#Phone"/>
        <owl:disjointWith rdf:resource="#Ticket"/>
        <owl:disjointWith rdf:resource="#Bottle"/>
        <owl:disjointWith rdf:resource="#Luggage"/>
        <owl:disjointWith rdf:resource="#Knife"/>
        <owl:disjointWith rdf:resource="#Book"/>
    </owl:Class>
    <owl:Class rdf:ID="Passenger">
        <rdfs:subClassOf rdf:resource="#Person"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasObject"/>
                <owl:someValuesFrom rdf:resource="#HumanObject"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Person">
        <rdfs:subClassOf rdf:resource="#DomainConcept"/>
    </owl:Class>
    <owl:Class rdf:ID="Phone">
        <rdfs:subClassOf rdf:resource="#HumanObject"/>
        <owl:disjointWith rdf:resource="#Toy"/>
        <owl:disjointWith rdf:resource="#Cigarette"/>
        <owl:disjointWith rdf:resource="#Bottle"/>
        <owl:disjointWith rdf:resource="#Luggage"/>
        <owl:disjointWith rdf:resource="#Ticket"/>
        <owl:disjointWith rdf:resource="#Paint"/>
        <owl:disjointWith rdf:resource="#Knife"/>
        <owl:disjointWith rdf:resource="#Book"/>
    </owl:Class>
    <owl:Class rdf:ID="Phone_Abuse">
        <rdfs:subClassOf rdf:resource="#Situation"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasObject"/>
```

```
            <owl:someValuesFrom rdf:resource="#Phone"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#involvedInSituation"/>
            <owl:someValuesFrom rdf:resource="#Aggressor"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#Smoking"/>
    <owl:disjointWith rdf:resource="#Abandoned_Luggage"/>
    <owl:disjointWith rdf:resource="#Vandalism"/>
    <owl:disjointWith rdf:resource="#Beggar"/>
    <owl:disjointWith rdf:resource="#Fainting"/>
    <owl:disjointWith rdf:resource="#Drunkard"/>
    <owl:disjointWith rdf:resource="#Theft"/>
    <owl:disjointWith rdf:resource="#Fighting"/>
    <owl:disjointWith rdf:resource="#Vomiting"/>
    <owl:disjointWith rdf:resource="#Graffiti"/>
    <owl:disjointWith rdf:resource="#Screaming"/>
</owl:Class>
<owl:Class rdf:ID="Scenario">
    <rdfs:subClassOf rdf:resource="#DomainConcept"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasSituation"/>
            <owl:someValuesFrom rdf:resource="#Situation"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Screaming">
    <rdfs:subClassOf rdf:resource="#Situation"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#involvedInSituation"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <owl:Class rdf:about="#Aggressor"/>
                        <owl:Class rdf:about="#Passenger"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
```

```
            </owl:Restriction>
        </rdfs:subClassOf>
        <owl:disjointWith rdf:resource="#Fighting"/>
        <owl:disjointWith rdf:resource="#Abandoned_Luggage"/>
        <owl:disjointWith rdf:resource="#Beggar"/>
        <owl:disjointWith rdf:resource="#Smoking"/>
        <owl:disjointWith rdf:resource="#Fainting"/>
        <owl:disjointWith rdf:resource="#Graffiti"/>
        <owl:disjointWith rdf:resource="#Vomiting"/>
        <owl:disjointWith rdf:resource="#Theft"/>
        <owl:disjointWith rdf:resource="#Phone_Abuse"/>
        <owl:disjointWith rdf:resource="#Drunkard"/>
        <owl:disjointWith rdf:resource="#Vandalism"/>
    </owl:Class>
    <owl:Class rdf:ID="Sickness">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasSituation"/>
                <owl:someValuesFrom>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                            <owl:Class rdf:about="#Fainting"/>
                            <owl:Class rdf:about="#Vomiting"/>
                        </owl:unionOf>
                    </owl:Class>
                </owl:someValuesFrom>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf rdf:resource="#Scenario"/>
        <owl:disjointWith rdf:resource="#Annoyance"/>
        <owl:disjointWith rdf:resource="#Danger"/>
        <owl:disjointWith rdf:resource="#Neutral"/>
        <owl:disjointWith rdf:resource="#Damage"/>
    </owl:Class>
    <owl:Class rdf:ID="Situation">
        <rdfs:subClassOf rdf:resource="#DomainConcept"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Observations from others</rdfs:comment>
    </owl:Class>
    <owl:Class rdf:ID="Smoking">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasObject"/>
```

```
            <owl:someValuesFrom rdf:resource="#Cigarette"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#involvedInSituation"/>
            <owl:someValuesFrom rdf:resource="#Aggressor"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#Situation"/>
    <owl:disjointWith rdf:resource="#Vandalism"/>
    <owl:disjointWith rdf:resource="#Theft"/>
    <owl:disjointWith rdf:resource="#Abandoned_Luggage"/>
    <owl:disjointWith rdf:resource="#Fainting"/>
    <owl:disjointWith rdf:resource="#Fighting"/>
    <owl:disjointWith rdf:resource="#Phone_Abuse"/>
    <owl:disjointWith rdf:resource="#Graffiti"/>
    <owl:disjointWith rdf:resource="#Vomiting"/>
    <owl:disjointWith rdf:resource="#Drunkard"/>
    <owl:disjointWith rdf:resource="#Screaming"/>
    <owl:disjointWith rdf:resource="#Beggar"/>
</owl:Class>
<owl:Class rdf:ID="Theft">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#involvedInSituation"/>
            <owl:someValuesFrom rdf:resource="#Aggressor"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#Situation"/>
    <owl:disjointWith rdf:resource="#Drunkard"/>
    <owl:disjointWith rdf:resource="#Fainting"/>
    <owl:disjointWith rdf:resource="#Screaming"/>
    <owl:disjointWith rdf:resource="#Phone_Abuse"/>
    <owl:disjointWith rdf:resource="#Fighting"/>
    <owl:disjointWith rdf:resource="#Graffiti"/>
    <owl:disjointWith rdf:resource="#Beggar"/>
    <owl:disjointWith rdf:resource="#Smoking"/>
    <owl:disjointWith rdf:resource="#Vandalism"/>
    <owl:disjointWith rdf:resource="#Abandoned_Luggage"/>
    <owl:disjointWith rdf:resource="#Vomiting"/>
</owl:Class>
<owl:Class rdf:ID="Ticket">
```

```
        <rdfs:subClassOf rdf:resource="#HumanObject"/>
        <owl:disjointWith rdf:resource="#Toy"/>
        <owl:disjointWith rdf:resource="#Bottle"/>
        <owl:disjointWith rdf:resource="#Knife"/>
        <owl:disjointWith rdf:resource="#Luggage"/>
        <owl:disjointWith rdf:resource="#Book"/>
        <owl:disjointWith rdf:resource="#Paint"/>
        <owl:disjointWith rdf:resource="#Phone"/>
        <owl:disjointWith rdf:resource="#Cigarette"/>
    </owl:Class>
    <owl:Class rdf:ID="Toy">
        <rdfs:subClassOf rdf:resource="#HumanObject"/>
        <owl:disjointWith rdf:resource="#Knife"/>
        <owl:disjointWith rdf:resource="#Cigarette"/>
        <owl:disjointWith rdf:resource="#Luggage"/>
        <owl:disjointWith rdf:resource="#Paint"/>
        <owl:disjointWith rdf:resource="#Ticket"/>
        <owl:disjointWith rdf:resource="#Bottle"/>
        <owl:disjointWith rdf:resource="#Phone"/>
        <owl:disjointWith rdf:resource="#Book"/>
    </owl:Class>
    <owl:Class rdf:ID="TrainCompartment">
        <rdfs:subClassOf rdf:resource="#DomainConcept"/>
    </owl:Class>
    <owl:Class rdf:ID="TrainInterior">
        <rdfs:subClassOf rdf:resource="#TrainObject"/>
        <owl:disjointWith rdf:resource="#TrainSeat"/>
        <owl:disjointWith rdf:resource="#TrainTable"/>
        <owl:disjointWith rdf:resource="#WalkPath"/>
    </owl:Class>
    <owl:Class rdf:ID="TrainObject">
        <rdfs:subClassOf rdf:resource="#DomainConcept"/>
    </owl:Class>
    <owl:Class rdf:ID="TrainSeat">
        <rdfs:subClassOf rdf:resource="#TrainObject"/>
        <owl:disjointWith rdf:resource="#TrainTable"/>
        <owl:disjointWith rdf:resource="#WalkPath"/>
        <owl:disjointWith rdf:resource="#TrainInterior"/>
    </owl:Class>
    <owl:Class rdf:ID="TrainTable">
        <rdfs:subClassOf rdf:resource="#TrainObject"/>
        <owl:disjointWith rdf:resource="#TrainSeat"/>
        <owl:disjointWith rdf:resource="#WalkPath"/>
```

```
    <owl:disjointWith rdf:resource="#TrainInterior"/>
</owl:Class>
<owl:Class rdf:ID="Vandalism">
    <rdfs:subClassOf rdf:resource="#Situation"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#involvedInSituation"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <owl:Class rdf:about="#Aggressor"/>
                        <owl:Class rdf:about="#TrainObject"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#Screaming"/>
    <owl:disjointWith rdf:resource="#Fighting"/>
    <owl:disjointWith rdf:resource="#Abandoned_Luggage"/>
    <owl:disjointWith rdf:resource="#Beggar"/>
    <owl:disjointWith rdf:resource="#Drunkard"/>
    <owl:disjointWith rdf:resource="#Fainting"/>
    <owl:disjointWith rdf:resource="#Theft"/>
    <owl:disjointWith rdf:resource="#Vomiting"/>
    <owl:disjointWith rdf:resource="#Phone_Abuse"/>
    <owl:disjointWith rdf:resource="#Graffiti"/>
    <owl:disjointWith rdf:resource="#Smoking"/>
</owl:Class>
<owl:Class rdf:ID="Vomiting">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#involvedInSituation"/>
            <owl:someValuesFrom rdf:resource="#Passenger"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#Situation"/>
    <owl:disjointWith rdf:resource="#Fainting"/>
    <owl:disjointWith rdf:resource="#Smoking"/>
    <owl:disjointWith rdf:resource="#Abandoned_Luggage"/>
    <owl:disjointWith rdf:resource="#Phone_Abuse"/>
    <owl:disjointWith rdf:resource="#Graffiti"/>
    <owl:disjointWith rdf:resource="#Screaming"/>
```

```
        <owl:disjointWith rdf:resource="#Fighting"/>
        <owl:disjointWith rdf:resource="#Theft"/>
        <owl:disjointWith rdf:resource="#Vandalism"/>
        <owl:disjointWith rdf:resource="#Drunkard"/>
        <owl:disjointWith rdf:resource="#Beggar"/>
    </owl:Class>
    <owl:Class rdf:ID="WalkPath">
        <rdfs:subClassOf rdf:resource="#TrainObject"/>
        <owl:disjointWith rdf:resource="#TrainSeat"/>
        <owl:disjointWith rdf:resource="#TrainTable"/>
        <owl:disjointWith rdf:resource="#TrainInterior"/>
    </owl:Class>
</rdf:RDF>
```

Appendix B

Survey Report

Analysis Report
(Aggression Survey)

By
V.Gangaram Panday
(January 2007)

Delft University of Technology
Faculty of Information Technology and Systems
Man Machine Interaction group

Table of Contents

1. Introduction

As part of my thesis work on aggression detection on trains a survey was done.
Our approach relies on having real-world knowledge about people's common attitudes towards situations, things, people, and actions. If we want our detection engine to be robust, we will have to supply it with knowledge that reflects this.
I will use this data later in my research to assign aggression ratings to scenarios in an automatic multimodal detection system.
The questions in this survey were in need of 2 types of answers. First we wanted to know how the respondent would feel given the situation described in the question on a scale from safe to danger (1=safe, 5=danger).And second how the respondent would experience the same situation on an active/neutral/passive scale. This combination will give us an idea on the aggressiveness of the described situation.
The questionnaire included a total of 38 questions (see Appendix) and was surveyed in November/December 2006 and in total 31 Computer Science students was interviewed.

2. Survey Setup

2.1 Goal of the Survey

Our approach relies on having real-world knowledge about people's common attitudes towards *situations, things, people*, and *actions*. If we want our detection engine to be robust, we will have to supply it with knowledge that reflects this.
The objective of this survey is to find out how safe people feel when being confronted with different situations while traveling in a train. Along with how they feel we want to know if they experience it actively, neutrally or passively. Ultimately the combination of these 2 variables will give an idea of the degree of aggression that will be given to the described situation. This information will be used when making decision rules later in my thesis work.

2.1 Target Group

We are interested in all the frequent or daily commuters that use the train to get them to work, school, etc. Students are a important part of this group as they use this method of transportation almost daily. So we choose to survey them about how they would act on different situations in the train, as they might have experienced some of the situations before.

2.2 Type of questions

In the appendix is a copy of all the questions in our survey. I tried to cover all sorts of aggressive and also more neutral questions. Control questions were also inserted to check if the respondent is really answering the questions honestly. Control questions are questions that we know they answer to beforehand and it would be the same for almost everyone. If these questions were not answered accordingly, then the survey of this particular respondent would be discarded.

2.3 How the results will be processed

In order to start analyzing the data gathered from the survey, I first need to identify a couple of important facts. As described in [Fink03a] these are:

- Dependent Variables
- Measurement Scales
- Statistical analysis methods

2.3.1 Dependent Variables

A variable in surveys is a characteristic that is measurable. The choice of analyzing survey data will be dependent on the type of data available and on the number of dependent variables involved. The dependent variables can be seen as the outcome or result of the survey. The dependent variables can be found by studying the survey's objectives and target. In this survey we deal with 2 dependent variables. These are "degree of safety" and "type of influence". Next step is to determine the measurement scale for these variables.

2.3.2 Measurement Scales

A characteristic may be surveyed and measured using a nominal, ordinal, or numerical scale and the resulting data are termed nominal, ordinal, or numerical.
In this survey an ordinal scale is used. Ordinal scales are typically used in questions that call for ratings of quality or agreement (like in this case safe/ probably safe/neutral/probably dangerous/dangerous and passive/neutral/active).
So our 2 variables contain ordinal data.

2.3.3 Statistical Analysis Methods

To determine a possible analysis method for our data again we look at the survey's objectives. In fact our survey is not too complex. We just need for every different situation (item or question) the most typical outcome for all the respondents. And we want that for both variables. In [Fink03b] it is discussed that when dealing with ordinal data and if you are concerned with the typical score then the use of a Median is a good predictor. Also it is discussed [Fink03a] that

ordinal data can be treated as numerical data if we assign a number to each rating measure of the ordinal data. That is what I have done too with my data.
Another interesting measure is the measure of dispersion (or spread). There are four types of measures of dispersion but only one is of interest to me and that is the standard deviation.
Another measure that gives a central tendency of an ordinally measured variable is the mode. The mode calculates the most frequently chosen output by all the respondents. Although the median give more information then the mode the latter will give a good overview if this corresponds to the median.
I have calculated the average too, but this isn't the best measure to interpret the data, because outlier data of the range can easily pull the average up or down.
But the combination of the average and the standard deviation will give an idea in which degree the average is a good predictor.

3. Data Entry

First all the data from the survey's were entered into an excel sheet. (This excel sheet will be added as an appendix). The table with data is big so in table 1 below only a part of the table with the raw data is presented as an illustration.

	RESPONDENTS															
Questions	1		2		3		4		5		6		7		8	
1	1	2	1	2	1	3	3	2	3	2	3	2	1	2	3	3
2	1	1	1	2	1	2	4	2	3	2	3	1	2	2	3	2
3	3	2	3	2	3	2	5	3	2	2	2	1	3	2	4	2
4	1	3	2	3	3	3	4	3	4	3	4	3	1	2	3	2
5	2	3	2	3	4	3	4	3	4	2	4	3	2	2	3	2
6	1	1	1	3	3	2	4	2	4	3	3	2	1	2	3	2
7	1	1	1	3	3	2	4	2	3	2	3	2	1	2	4	2
8	1	3	1	3	1	3	1	1	3	2	3	2	1	1	4	3
9	2	2	1	3	3	1	4	2	4	3	2	1	1	1	4	2
10	1	1	1	2	1	1	3	2	5	2	4	2	1	1	3	2
11	1	1	1	1	1	1	2	2	1	1	3	2	1	1	3	2
12	1	1	1	1	1	2	3	3	4	3	4	2	1	1	3	2
13	1	1	1	1	1	1	2	1	3	2	3	2	1	1	3	2
14	1	2	1	2	1	1	3	1	3	2	3	2	1	1	4	2
15	1	2	1	3	1	3	3	1	3	2	3	2	1	1	4	3
16	1	1	1	1	1	1	3	2	3	1	3	2	1	1	3	2
17	3	2	2	2	4	3	4	3	5	3	3	2	3	1	4	3
18	4	3	4	3	3	2	4	3	4	3	3	2	2	1	4	3
19	1	2	1	2	1	1	3	2	3	2	1	3	1	1	4	3
20	2	3	1	3	4	3	3	2	4	3	2	2	1	1	5	3
21	1	2	1	2	1	1	3	2	3	2	3	2	1	1	4	2
22	1	2	1	3	3	2	4	2	5	3	4	2	1	1	4	2
23	1	2	1	1	1	1	2	2	3	2	3	2	1	1	3	2
24	1	3	1	3	1	3	2	3	3	2	1	3	1	1	4	2
25	3	1	3	2	1	1	5	1	3	2	3	2	3	2	5	3
26	1	1	1	1	3	2	4	1	5	3	1	2	1	1	3	2
27	1	2	2	2	1	2	4	3	5	3	2	2	1	1	4	2
28	5	3	5	3	5	3	5	3	5	3	4	1	4	3	5	3
29	1	1	4	3	3	1	4	2	4	2	3	1	3	2	5	3

30	1	1	3	3	1	3	4	3	3	2	3	2	2	1	3	2
31	1	2	2	3	1	3	4	2	4	2	3	1	1	1	3	2
32	1	1	1	1	1	1	2	2	5	2	3	2	1	1	3	2
33	1	2	1	1	1	1	2	2	3	2	3	2	1	1	3	2
34	1	2	1	3	2	3	3	3	3	2	3	1	1	1	4	2
35	1	1	1	1	1	1	3	2	3	3	3	1	1	1	3	2
36	3	2	3	3	4	1	5	3	5	3	4	1	2	2	4	2
37	5	3	5	3	4	3	5	3	5	3	5	1	5	3	5	3
38	5	3	5	3	5	3	5	3	5	3	5	1	4	2	5	3

Table1. Raw Data

The table shows in the first column all the 38 question of the survey and the first row give all the respondents. Per respondent there are two columns which correspond to the two variables we are interested in. The first column per respondent gives the “degree of safety” and the second column is for the “degree of influence”.

To apply the different analysis methods on the two variables table 1 was divided into 2 tables displaying only the data for the two variables. One table had all the data for the “degree of safety” variable which respondents answered to each question. And the other table had all the data for the “degree of influence” variable. The table below gives an illustration of how the data for the first variable was entered. (The complete table can be found in the excel sheet).

	Respondents													
Questions	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	1	1	1	3	3	3	1	3	1	1	1	1	1	1
2	1	1	1	4	3	3	2	3	1	1	1	2	2	1
3	3	3	3	5	2	2	3	4	1	4	3	3	2	2
4	1	2	3	4	4	4	1	3	1	4	2	3	2	2
5	2	2	4	4	4	4	2	3	3	4	2	4	3	2
6	1	1	3	4	4	3	1	3	3	3	2	2	3	1
7	1	1	3	4	3	3	1	4	2	3	3	3	3	1
8	1	1	1	1	3	3	1	4	1	1	1	1	1	1
9	2	1	3	4	4	2	1	4	3	3	3	2	1	1
10	1	1	1	3	5	4	1	3	1	2	2	1	1	1
11	1	1	1	2	1	3	1	3	1	1	1	1	1	1

4. Results

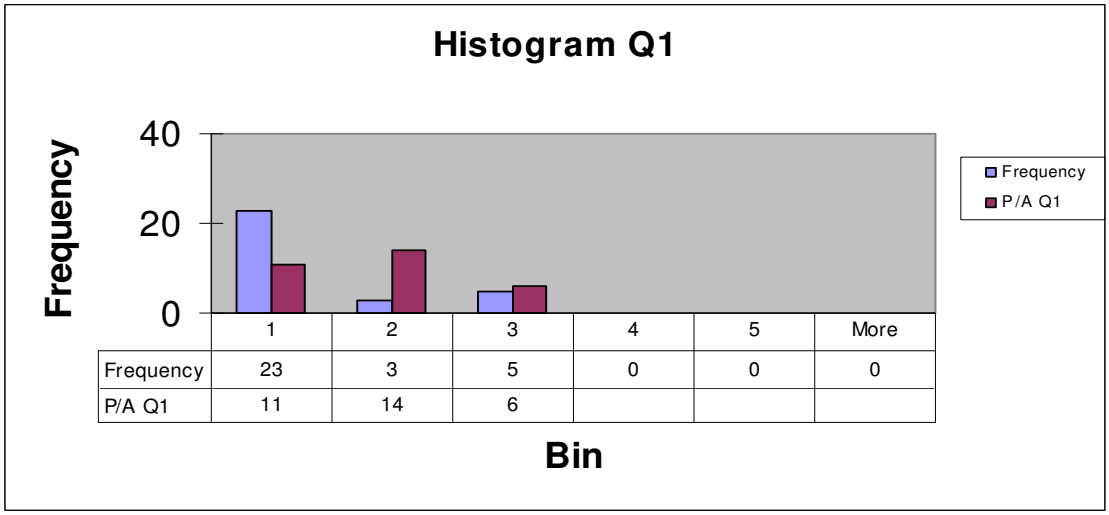
Questions	Safe-danger scale				Passive-active scale			
	MEDIAN	STD	Average	Mode	MEDIAN	STD	Average	Mode
1	1	0.765	1.42	1	2	0.735	1.84	2
2	1	0.945	1.68	1	2	0.558	1.61	2
3	3	1.194	2.68	3	2	0.601	1.81	2
4	3	1.180	2.52	3	2	0.709	2.35	3
5	3	1.031	2.94	3	2	0.568	2.45	3
6	3	0.996	2.52	3	2	0.619	1.87	2
7	3	1.092	2.52	3	2	0.597	1.90	2
8	1	0.791	1.32	1	1	0.803	1.61	1
9	3	1.057	2.42	3	2	0.746	1.90	2
10	1	1.077	1.68	1	1	0.677	1.52	1
11	1	0.543	1.19	1	1	0.599	1.32	1
12	1	1.006	1.71	1	2	0.709	1.65	1
13	1	0.761	1.39	1	1	0.461	1.29	1
14	1	0.882	1.61	1	2	0.615	1.61	2
15	2	1.264	2.26	1	2	0.772	2.06	2
16	1	0.807	1.42	1	1	0.541	1.32	1
17	3	1.077	3.32	3	2	0.772	2.06	2
18	4	0.811	3.52	4	3	0.720	2.42	3
19	1	0.890	1.52	1	2	0.730	2.00	2
20	2	1.226	2.35	1	2	0.727	2.06	2
21	1	0.985	1.65	1	2	0.620	1.58	1
22	2	1.235	2.48	1	2	0.746	1.90	2
23	1	0.769	1.52	1	1	0.624	1.55	1
24	2	0.934	1.84	1	3	0.768	2.45	3
25	4	1.288	3.48	3	2	0.700	2.10	2
26	1	1.054	1.61	1	1	0.624	1.45	1
27	2	1.155	2.00	1	2	0.631	1.74	2
28	4	0.871	4.32	5	3	0.909	2.32	3
29	3	1.204	3.13	3	2	0.727	1.94	2
30	3	1.148	2.42	3	2	0.727	2.06	2
31	2	1.031	1.94	1	2	0.752	1.97	2
32	1	0.962	1.48	1	1	0.541	1.32	1
33	1	0.693	1.29	1	1	0.551	1.35	1
34	2	1.014	2.19	2	2	0.772	2.06	2
35	1	0.810	1.45	1	1	0.608	1.35	1

36	3	1.226	3.35	3	2	0.653	2.32	2
37	5	0.341	4.87	5	3	0.715	2.61	3
38	5	0.428	4.87	5	3	0.672	2.58	3

The above tables show the calculated measures, median, Standard deviation, Average and mode of both variables per question. For example for question 1 the central tendency for the first variable is that in the described situation people tend to feel safe (median is 1). And for the second variable the tendency goes towards feeling neutral (median is 2) in the given situation. Now if we look at the described situation in question 1: "Ticket inspection by the conductor" the result outcome seems acceptable as there would be no need for danger also because you have a valid ticket.

In general the mode measure follows the median nicely. This means that the most frequently chosen answer corresponds with the median. For the scale of the first variable ("degree of safety") 80% of the answered questions, the median and the mode are the same. For the other 20% the mode differs slightly from the mean. For the second variable ("degree of influence") at 90% of the answered questions have the same median and mode. Again the other 10% the mode differs just slightly.

We can take a deeper look at the results for question 1 by making a histogram of the respondents' answers.



Bin	*Frequency*	*%*
1	23	74.19355
2	3	9.677419
3	5	16.12903
4	0	0
5	0	0
More	0	

Bin	*Frequency*	*%*
1	11	35.48387
2	14	45.16129
3	6	19.35484
More	0	

This histogram shows for both variables how the answers were given to question one. The table below the histogram is just a tabular form of the histogram for both variables separately. In the histogram the blue bars give the answers given for the first variable and the purple bars for the second variable.

We see that 23 respondents answered 1 (=safe) to this question and 5 respondents answered 3 (=neutral). In percentages almost 75% of the respondent felt safe in this situation and 16% neutral.

For the second variable 45% of the respondents would experience the described situation as neutral and 35% as passive (e.g. not reacting visibly to the situation).

In this case these two answers can be considered the same as it is a matter of interpretation by the respondent.

In the same way all 38 question have been analyzed and of each question a detailed histogram as shown above has been made. These can be found in the excel sheet.

5. Conclusion

The results have been carefully studied and the outcome for each question is the same (or close to) my personal opinion to the questions in the survey. In fact these results give a clear and general view of how each of the situations is observed by the public.
It is necessary and useful to have the opinion of daily train users to cross-reference my opinion with that of the respondents.
For my further thesis study this data will be used to categorize these situations as a degree of aggression. With these results I can give a more accurate aggression label to a situation.

6. References

[Fink03a] A. Fink, **'How to manage, analyze, and interpret survey data'**, *Sage Publication, Inc.* 2003.

[Fink03b] A. Fink, **'The Survey Handbook'**, *Sage Publication Inc,* 2003.

7. Appendix

Example of the Survey

Aggression Scale Survey

This survey is designed to assess the impact of events in a train on social safety and satisfaction. Your responses will be used to improve and guide the development of an automated aggression detection system in the train. Responses will be kept anonymous.

Please imagine the following situation when you are answering the questions in this survey.

Your situation:

"You are sitting in a train, you have an appointment, but with plenty of time so not particularly in a hurry. The train was on time. You have a valid ticket"

For each of the following items we want you to identify the following two things which relate to each other:

1. How would you rate the situation described by each item in terms of danger.
2. How would rate your own experience (feeling) of the situation as you rated above.

For example:
A passenger sitting next to you starts to waive a knife at the conductor who wants to inspect his ticket.
In this example you might rate the situation as 5 (danger) and maybe because it's happening right in front of you, you might actively experience the danger. So you would choose 5(danger)/3(active).

1st row (1= safe ,3= neutral, 5= danger)
2nd row(1=passive , 2=neutral, 3= active)

Description/Identification of Survey Item	SCALE				
1. Ticket inspection by the conductor.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)

2.	Conductor issuing a fine to someone not having a ticket.	1 (safe)	2	3 (neutral)	4	5 (danger)
		1 (passive)		2 (neutral)		3 (active)
3.	Passenger heavily arguing with the conductor about not having a ticket	1 (safe)	2	3 (neutral)	4	5 (danger)
		1 (passive)		2 (neutral)		3 (active)
4.	A smoking passenger sitting across you.	1 (safe)	2	3 (neutral)	4	5 (danger)
		1 (passive)		2 (neutral)		3 (active)
5.	Conductor asks a smoking passenger to stop smoking as this is a non-smoking train, but once the conductor moves on the passenger starts smoking again.	1 (safe)	2	3 (neutral)	4	5 (danger)
		1 (passive)		2 (neutral)		3 (active)
6.	Beggar walking thru the train begging.	1 (safe)	2	3 (neutral)	4	5 (danger)
		1 (passive)		2 (neutral)		3 (active)
7.	Music playing beggar first plays a song in a full train and then comes to collect.	1 (safe)	2	3 (neutral)	4	5 (danger)
		1 (passive)		2 (neutral)		3 (active)
8.	Announcement through the speakers that the next station will be Delft CS.	1 (safe)	2	3 (neutral)	4	5 (danger)
		1 (passive)		2 (neutral)		3 (active)
9.	Group of youngsters disturbing the order by talking loudly, laughing and yelling at each other.	1 (safe)	2	3 (neutral)	4	5 (danger)
		1 (passive)		2 (neutral)		3 (active)
10.	Group of youngster just playing a fun card game across you and occasionally talking a bit loudly.	1 (safe)	2	3 (neutral)	4	5 (danger)
		1 (passive)		2 (neutral)		3 (active)
11.	Two Passengers across you normally talking to each other about work related issues in a crowded train.	1 (safe)	2	3 (neutral)	4	5 (danger)
		1 (passive)		2 (neutral)		3 (active)

12. Passenger putting his/her feet on the seats.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
13. Passenger talking normally on the phone with a colleague telling her she will be a little bit later.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
14. Passenger talking loudly on his/her mobile phone with a friend about how great the weekend was.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
15. Train stops and there is an announcement that due to an accident ahead this train will go back to the last station	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
16. Passenger talking business on his mobile phone on a normal volume level in a crowded train.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
17. Passenger sketching graffiti on the empty trains' interior at night.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
18. Passenger drawing with a pen on the trains' seat across you and when you say something about it the passenger just ignores you.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
19. Mother with a baby trolley trying to find a spot in a full train.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
20. Loud rap music playing by a passenger 2 rows further in a crowded train.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
21. Passenger playing techno music on a silent volume level right next to you which u can hear thru his earphone.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)

22. Passenger playing loud techno music on his mobile phone speakers a couple of rows away from you.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
23. Family that goes on vacation with lots of baggage tries to find a place to sit in a full train	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
24. You are trying to find a seat and ask a passenger to remove their bag from the seat in a full train.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
25. Announcement through the speakers that the train won't stop at Schiphol because of a possible bomb threat.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
26. Passenger with a bicycle enters a empty train at night	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
27. Passenger tries to enter the train at rush hour with a bicycle	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
28. Two people fighting in an empty train at night.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
29. A passenger with a non-valid ticket does not want to cooperate with the conductor asking for his identification.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
30. Passenger trying to make conversation with you in an empty train.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
31. Passenger trying to make conversation with you in crowded train.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)

32. Passenger that come sit next to you takes out a pile of papers and start browsing through them.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
33. Passenger with a laptop doing some typing while sitting next to you in a full train.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
34. Passenger sitting next to you takes out a big file from his bag and when opened takes a lot of space from you.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
35. Passenger browsing thru the newspaper while sitting next to you.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
36. A drunken passenger talking rubbish and sitting across you in a full train.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
37. Passenger with a gun starts to threaten the conductor when he wants to inspect his ticket in an empty train.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)
38. Two supporter groups enter a empty train after a football match and start to fight with each other destroying all the interior.	1 (safe)	2	3 (neutral)	4	5 (danger)
	1 (passive)		2 (neutral)		3 (active)

Appendix C

Restriction Types

The commonly used restriction is the quantifier restrictions. These types of restrictions are composed of a *quantifier*, a property, and a *filler*. The two quantifiers that may be used are:

- The *existential quantifier* ($\exists$), which can be read as at *least one, or some*[1].
- The *universal quantifier* ($\forall$), which can be read as *only*[2].

All types of restrictions describe an unnamed set that could contain some individuals. This set can be thought of as an anonymous class. Any individuals that are members of this anonymous class satisfy the restriction that describes the class (Figure C.1). Restrictions describe the constraints on relationships that the individuals participate in for a given property. When we describe a named class using restrictions, what we are effectively doing is describing anonymous super-classes of the named class.

[1] It can also be read as 'someValuesFrom' in OWL speak.
[2] It can also be read as 'allValuesFrom' in OWL speak.

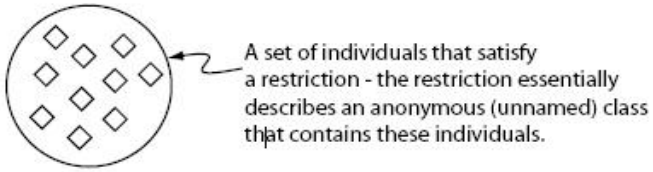


Figure C.1: Restrictions describe Anonymous classes of Individuals.

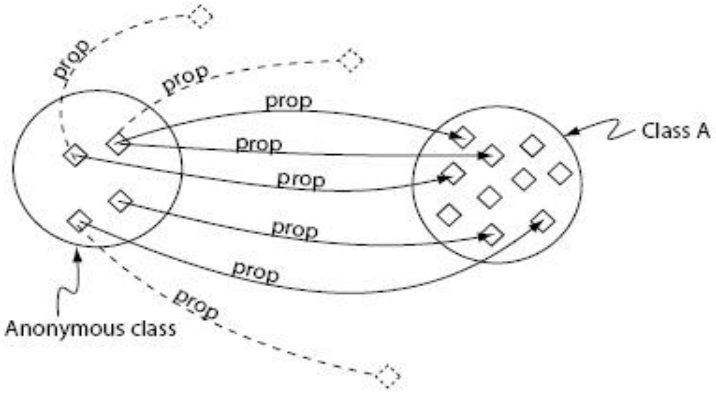


Figure C.2: A Schematic Of An Existential Restriction (∃ **prop ClassA**).

Existential restrictions, also known as 'someValuesFrom' restrictions, or 'some' restrictions are denoted using ∃. Existential restrictions describe the set of individuals that have at least one specific kind of relationship to individuals that are members of a specific class. Figure C.2 shows an abstracted schematic view of an existential restriction, ∃ **prop ClassA** i.e. a restriction along the property **prop** with a filler of **ClassA**. Notice that all the individuals in the anonymous class that the restriction defines have at least one relationship along the property **prop** to an individual that is a member of the class **ClassA**. The dashed lines in Figure C.2 represent the fact that the individuals may have other **prop** relationships with other individuals that are not members of the class **ClassA** even though this has not been explicitly stated The existential restriction does not constrain the prop relationship to members of the class **ClassA**, it just states that every individual must have at least one **prop** relationship with a member of **ClassA** this is the *open world assumption*(OWA).

Universal restrictions are also known as 'allValuesFrom' restrictions, or 'All' restrictions since they con- strain the filler for a given property to a specific class. Universal restrictions are given the symbol ∀. Universal restrictions describe the set of individuals that, for a given property, only have relationships to other individuals that are members of a specific class. A feature of universal restrictions, is that for the given property, the set of individuals that the restriction describes will also contain the individuals that do not have any relationship along this property to any other individuals. A universal restriction

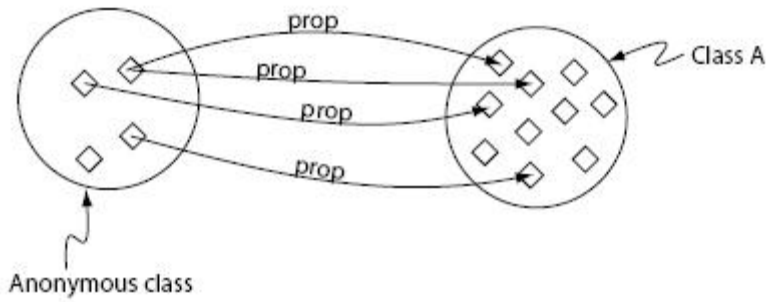


Figure C.3: A Schematic Of An Universal Restriction (∀ **prop ClassA**).

along the property prop with a filler of **ClassA** is depicted in Figure C.3. Once again, an important point to note is that universal restrictions do not 'guarantee' the existence of a relationship for a given property. They merely state that if such a relationship for the given property exists, then it must be with an individual that is a member of a specified class.

A **hasValue** restriction, denoted by the symbol ∋, describes an anonymous class of individuals that are related to another specific individual along a specified property. Contrast this with a quantifier restriction where the individuals that are described by the quantifier restriction are related to any individual from a specified class along a specified property. Figure C.4 shows a schematic view of the hasValue restriction **prop** ∋ **abc**. This restriction describes the anonymous class of individuals that have at least one relationship along the prop property to the specific individual **abc**. The dashed lines in Figure C.4 represent the fact that for a given individual the hasValue restriction does not constrain the property used in the restriction to a relationship with the individual used in the restriction i.e. there could be other relationships along the **prop** property. It should be noted that hasValue restrictions are semantically equivalent to an existential restriction along the same property as the hasValue restriction, which has a filler that is an enumerated class that contains the individual (and only the individual) used in the hasValue restriction.

At last you have the cardinality restrictions. **Cardinality** restrictions are used to talk about the number of relationships that an individual may participate in for a given property. Cardinality restrictions are conceptually easier to understand than quantifier restrictions, and come in three flavors: Minimum cardinality restrictions, Maximum cardinality restrictions, and Cardinality restrictions.

Minimum cardinality restrictions specify the minimum number of relationships that an individual must participate in for a given property. The symbol for a minimum cardinality restriction is the 'greater than or equal to' symbol (≥). Minimum cardinality restrictions place no maximum limit on the number of relationships that an individual can participate in for a given

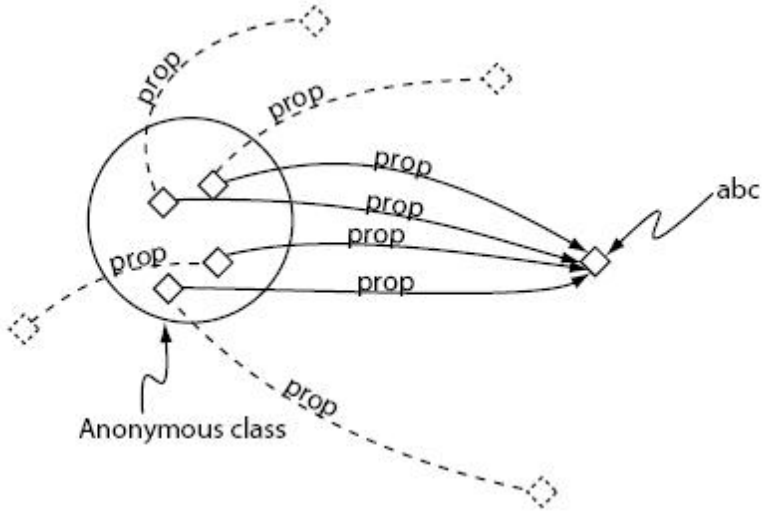


Figure C.4: A Schematic View Of The hasValue Restriction, **prop** ∋ **abc** - dashed lines indicate that this type of restriction does not constrain the property used in the hasValue restriction solely to the individual used in the hasValue restriction.

property.

Maximum cardinality restrictions specify the maximum number of relationships that an individual can participate in for a given property. The symbol for maximum cardinality restrictions is the 'less than or equal to' symbol ($\leq$). Note that maximum cardinality restrictions place no minimum limit on the number of relationships that an individual must participate in for a specific property.