Utrecht University
Graduate School of Natural Sciences
Master Artificial Intelligence

ORACLE: An Ontology Reasoner for Affective Conversation in Long-term Engagement

Author:
Jan de Mooij
3966615

22 June 2018

Daily Supervisor:
ir. M. van Bekkum
TNO

First Supervisor
dr. R. Iemhoff
Utrecht University

Second Examiner:
dr. C.P. Janssen
Utrecht University

Abstract

When social robots interact with the same user for longer periods of time, memory can increase the personalization of that robot. In this research, episodic memory in long term interaction for social robots is investigated. Theories on human memory and question asking were combined with those from the field of social artificial agents, which resulted in an ontology where events from the user's life can be encoded in a 5W1H (who/what/where/when/why/how) structure. This allows a dialog manager to form personalized open questions on properties missing in that encoded structure. Furthermore, the model allows inferring information missing from properties – or find contradicting information in properties – on similar episodes, which can be used to further personalize questions. Lastly, concepts of affect, such as sentiment, emotion, and mood, were added so the affective state of the user can be taken into account during question formulation.

keywords: Episodic memory; Question asking; Affect; Social robot; Event representation; Symbolic pattern recognition

Acknowledgements

This research project and thesis describing it have slowly come to life over the course of seven months. Although I have enjoyed most of the process, there have been difficult moments, where others have helped me move forward.

First of all, I'd like to thank my supervisors. Michael, who not only helped me along the way during our weekly meetings, but also knew when encouraging words were necessary and shares my interest in well-used sarcasm. Rosalie, my first supervisor at the Utrecht University, who showed enthusiasm from start to end, and agreed to supervise me because she believed in me, despite no formal logic being involved. Chris, who, as my second examiner, usually would not be involved until the very end, but offered to step in when Rosalie had to take a step back, and was on top of what was still needed right away. Thank you, Michael, Rosalie and Chris.

I also want to thank Peter, a friendly neighbor and colleague both at the master's program, and at TNO, where we both participated in the PAL project. Thank you Peter, by leading by example on the balance between work and serious games of table tennis, and thank you for the valuable discussions relating to the PAL project and also on unrelated philosophies.

I want to further thank everybody who made our hall at TNO the most fun to be at. My co-interns who made me know I was not going through the process alone and were always available with the right input when an opinion or joke was needed. Also everybody from the Connected Business group at TNO, who show a work environment can be productive *and* a good place for fun, and never hesitated to share in their rich knowledge when asked.

Last, but not least, I want to thank my friends Dignee and Sil, who provided the mental distractions I needed when I was off-duty. Spending my evenings with either of you always gave me much needed energy, and allowed me to go on for as long as I have. Thank you both.

This work was partly conducted in the context PAL project, which was funded by the Horizon 2020 program, ref. H2020-PHC-643783. This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

List of abbreviations

5W1H Who, What, Where, When, Why, How (sometimes including sentiment)

EMM Episodic Memory Manager - Episodic Memory toolkit by Schreuder Goedheijt (2017)

EPMEM Episodic Memory - abbreviation often used in computer science implementations

ESMAS Emotional State Model for Affective Semantics

HPC Health Care Professional

NLP Natural Language Processing

NLU Natural Language Understanding

ORACLE Ontology Reasoner for Affective Conversation in Long-term Engagement

OWL Web Ontology Language

PAL Personal Assistant for a Healthy Lifestyle

RDF Resource Description Framework

URI Unique Resource Identifier

Contents

Chapter 1
Introduction

The Personal Assistant for a healthy Lifestyle (PAL) project is a research program, which aims to train, improve, and stimulate the demanding self-management skills of children with Type 1 Diabetes Mellitus (T1DM) between the ages of 8 and 14 years (Looije, 2015; Looije et al., 2016). Children in this age range with T1DM often depend heavily on their parents for their diabetes management. Not only is this demanding for those parents, but there exists a risk that when those children hit puberty, and avert themselves more from their parents, they will ignore their diabetes as well. Left unmanaged diabetes can cause serious damage to organs in the long run.

The PAL project uses a Social Robot, Charlie, who can become the child's "pal". Children are introduced to Charlie during a first hospital visit, where they play games with the robot and decide, together with their parents and a health care professional, which learning goals to pursue during a three to four month period. After this initial visit, they can talk with Charlie, and play games with him, through an application called MyPAL (Ligthart, 2016). This application contains a virtual avatar representation of Charlie, three games (a quiz, a memory game and a sorting game), and a diary, in which children can keep track of activities, the number of carbohydrates consumed during meals and snacks, and blood glucose measurements. The robot's avatar can provide feedback on e.g. items entered in the diary (or lack thereof), progression of the child towards goals and how well a child scored in a game they played with or against the robot (Van Stee, 2017).

1.1 Problem Statement

The effectiveness of the PAL program hinges directly on the usage patterns of the children. To effect behavior change, users need to stick with a self-management program for a longer period of time. Within the PAL project, where children can make use of the application over a period of three to four months, a gradual decrease in motivation to use the MyPal application can be seen with participants,

which can likely be attributed to the novelty effect wearing off (Schreuder Goedheijt, 2017; Henkemans et al., 2017). The novelty effect occurs when people are acquainted with something new, that, because of its novelty, becomes interesting. Once the 'new' starts to become familiar, this interest gradually wears off. At that point, the product itself, rather than its novelty, must be interesting to stop users from quitting the product, or the user should otherwise be motivated to continue using it. Henkemans et al. (2017) found that personalizing the robot increases affection from the child, and improves the child's motivation to play the quiz game. Henkemans et al. further found the wear-off of the novelty effect can be delayed this way, but were not able to show the novelty effect wear-off was prevented indefinitely.

A large part of the research within the PAL project is directed at the question how children can be kept motivated to use the MyPAL application. This is mainly done through supporting the children's innate needs of *autonomy*, *relatedness* and *competence*, which, according to the Self-Determination Theory (Deci and Ryan, 1985; Ryan and Deci, 2000), support the intrinsic motivation of the children.

1.2 Purpose and Goals

In previous research, an *episodic memory* component has been developed for the PAL platform (Schreuder Goedheijt, 2017). This module allows the PAL actor to remember and comment on experiences that it shares with the child. Specifically, the episodic memory component allows the PAL actor to comment on events recorded during previous interactions of the child with the MyPal application. These comments are currently limited to goal progression and glucose measurements. A comment on goal progression can be either to motivate the child to finish the goal if it had not yet been finished during the last interaction, or to start working towards a new goal if it had. Comments on glucose measurements relate to what the child has entered in the diary. For example, the PAL actor can ask the child to enter measurements from the previous day if those are missing in the diary or compliment a child for having blood glucose values in safe ranges for multiple days in a row.

Currently, the child is invited to enter open text to tell Charlie about themselves only at a few specific moments during interaction, while open text entries of children could potentially provide important insights in the child's progress and wellbeing. Furthermore, a corpus of open text entered by children within the PAL project could be useful for further research, such as training of sentiment and topic miners.

The purpose of this study is to investigate how the system's episodic memory can be utilized to ask children personalized open questions.

In order to ask the correct questions, the PAL actor should make use of information about the *sentiment* of an episode it has at its disposal. The sentiment of an episode indicates how the child felt when the episode happened, and is known to the system in two ways. First, when a child enters information in the diary, they can add their sentiment to that entry by using a slider. Secondly, a *sentiment mining* module has been added to the PAL system recently, which can annotate strings of natural language with the sentiments *positive*, *negative* or *neutral*.

1.3 Research Question

The research question asked in this project is the following:

RQ How can episodes and sentiments in the PAL system be used to ask personalized open questions?

The aim of this question is to investigate how the PAL actor can ask the child open questions which relate to the child's own experiences or feelings towards those experiences.

The episodes the robot chooses to comment on, and the time it chooses to do so, should work to enhance the relatedness between the child and the agent. Some episodes may not be relevant for the PAL actor, which concerns itself with the self-management skills of the child relating to T1DM. Further more, the PAL actor must understand an episode to some extent to be able to ask a meaningful question. This means the episodes which are used by the PAL system may be limited to only those which can be interpreted by the PAL actor. To this end, the first subquestion is:

SQ1 What types of episodes are useful for the PAL actor?

Episodes are contextualized by *topics*. A topic describes what the episode is about. During usage of the MyPal application, this will usually describe the activity of the child at the time of the episode, e.g. playing a quiz or adding an entry to the diary. An episode may also contain some other meta data, such as sentiment, a set of smaller activities within the episode, the exact form field the child was entering when the episode occurred or how long the episode lasted. However, an episode might not be completely annotated. In fact, there might be instances where an episode does not contain any meta data or contextual information at all. The second subquestion is:

SQ2 How can the PAL system work with episodes with incomplete annotation?

One aim of asking open questions related to episodes in the memory of the PAL System is to make dialogue appear more natural, which should increase the perceived intelligence of the PAL actor and the relatedness between the child and the PAL actor. To make conversation meaningful, however, the robot should be able to ask a follow-up question based on the answer of the child, so the conversation does not terminate after the child answers. To this end, the third subquestion is:

SQ3 How can a follow-up question be asked after an answer is given?

Since the PAL project uses the Web Ontology Language (OWL) as declarative memory, the results of previous subquestions may require some adjustments and additions to the ontology used by PAL. The third and final subquestion for the design part of the research question is therefor:

SQ4 How should the existing ontology be extended or adapted?

1.4 Structure of this thesis

In the next chapter background information to the current research will be provided by means of the literature, and related work will be discussed. The chapter starts with an overview of the PAL project, its research aims, and the structure of the accompanying system. Since the PAL project uses ontologies to store – and reason over – almost all persistent data, the concept of ontologies will subsequently be discussed. After that, recent research within the PAL project on the concepts of affect, mood, and emotion, as well as the resulting model will be discussed. These concepts are important for the current research efforts to personalize open questions. Lastly, theories from the fields of psychology, computer science and artificial intelligence regarding episodic memory will be discussed. The component developed by Schreuder Goedheijt (2017) was inspired by this memory system, and revisiting these theories might provide new inspiration for additional functionality required to answer the questions for the current research.

After the literature is in place, a more formal specification for the current research will be provided in Chapter 3 on page 28. An analysis of how the provided literature can inspire the current research is given, after which the method will be discussed.

In Chapter 4 on page 40, the implementation of the model satisfying the design criteria specified in the preceding chapter will be described, after which the model is validated using the approach given in the Section 3.3 on page 33 of Chapter 3.

Chapter 2
Background

2.1 Personal Assistant for a healthy Lifestyle (PAL)

The Personal Assistant for a healthy Lifestyle (PAL) project is a research project which investigates the role social robots[1] and mobile E-Health[2] applications can play in the teaching of self-management. The PAL project is specifically aimed at children with Type 1 Diabetes Mellitus (T1DM) between the ages of 8 and 14 years (Looije, 2015; Looije et al., 2016) and is funded under the Horizon 2020 program (ref. H2020-PHC-643783).

2.1.1 Type 1 Diabetes Mellitus

T1DM is a chronic auto-immune disease that attacks blood cells in the pancreas; the organ which is responsible for creating the hormone insulin that regulates glucose levels in the blood (World Health Organization, 2017). The number of T1DM diagnoses has risen significantly over the past few years (International Diabetes Federation, 2017; World Health Organization, 2017). The effect of T1DM, if unmanaged, is a raised blood glucose level – called *hyperglycemia* or *hyper* for short – or a blood glucose level that is too low – called *hypoglycemia* or *hypo* for short. Too much variation in these glucose levels can cause serious organ damage in the long run. However, with proper self-management, these levels can be kept in safe ranges (International Diabetes Federation, 2017). In the case of a hyper, this is done by manually administering additional insulin to the body, while in the case of a hypo eating a dextrose tablet or any other form of quickly released glucose

[1] A social robot is a robot that fulfills a social role. Instead of doing physical work, such as constructing products at an assembly line or helping in the house hold, social robot's primary objective is to be a social partner to a human being, which requires some sort of personality and understanding of emotional context. See e.g. Duffy (2003)

[2] E-Health is health care provided through digital means. In the case of PAL, health care is provided through a mobile application. This specific case of E-Health is sometimes called *M-Health*.

can raise blood glucose levels to within safe values.

In order to perform the proper glucose correction, one needs to know the blood glucose levels at that time. A diabetes patient measures this multiple times a day at fixed intervals but also e.g. after a meal or before physical exercise. Furthermore, a diabetes patient needs to keep track of the amount of carbohydrates they consume, as this influences the amount and type of insulin required for a correction. Because of this, correctly managing T1DM is an intensive effort, which requires rigorous self-management (Castensøe-Seidenfaden et al., 2017).

Typically, children in the ages of 8 to 14 years heavily depend on their parents for their diabetes management, which in turn is very demanding on the parents. Furthermore, once these children reach puberty, they tend to distance themselves from their parents and by extension, if they are still dependent on their parents for their diabetes management, they are more likely to ignore the importance of their diabetes management as well.

PAL tries to teach children the required self-management skills before they reach puberty for exactly this reason.

2.1.2 The PAL approach

The PAL project is a consortium of partners from The Netherlands, Italy, Germany and the United Kingdom. Partners are hospitals (2 based in The Netherlands, 1 in Italy), Universities, Research Institutes and private companies. TNO – the Dutch Organisation for Applied Sciences – acts as the project coordinator and manager[3].

During an experiment phase of the PAL Project, data is gathered about how children interact with an Aldebaran Nao robot and with a corresponding application (see Section 2.1.3 on the next page). Children are recruited in the participating hospitals. Children can be recruited if they are between 8 and 14 years old, and have been diagnosed with T1DM at least one and a half years prior to the start of the experiment (Looije et al., 2016).

Children who participate in the PAL project are acquainted with the robot, called Charlie, during a hospital visit. Charlie is programmed to act as the children's *pal*, helping them learn about T1DM and motivating them to adhere to the self-management they should learn. The robot itself is in no way intended to be an expert within the diabetes domain, and neither should it act as a teacher. While the robot itself is no expert, however, expert knowledge is present within the PAL system, and – through e.g. remarks from Charlie or quiz questions – disclosed to the child. This expert knowledge is always verified by professionals within the domain.

[3]For a complete description of the partners and their roles within the PAL consortium, see http://www.pal4u.eu/index.php/partners-2/

When the children meet Charlie, they can talk with him and play two games against the robot on a tablet standing between them. One of these games is a quiz with T1DM related questions (Kruijff-Korbayová et al., 2015). Child and robot take turns asking the questions, so the other person can answer. The other game is called collaborative sorting game, inspired by the 'Sandbox' technique, where boxes need to be broken open with the correct tools, revealing images of food which then need to be dragged to the correct category (Baxter et al., 2012).

During the same session, the child and their parents talk with a health care professional (HPC) to decide what self-management skills the child wants to learn during participation to the PAL project. These are expressed as *goals* (e.g. "go to birthday party alone", "know how to measure", "stay over at family alone", etc), which can be achieved by performing *tasks* (e.g. answering related quiz questions, watching a video, measure blood glucose levels independently, etc) (Peters et al., 2017).

2.1.3 MyPal

After the initial meeting with the robot, children participate for another 3 to 4 months. At home, they do not have a physical robot. Instead, they take an Android tablet home, with the MyPAL application pre-installed. This application contains a virtual avatar of the robot they met in the hospital (see Figure 2.1). This virtual avatar represents the same agent as the the physical robot, and when anything applies to both the physical robot and its digital avatar, both are together referred to as the PAL actor.

All games children play in the hospital are also integrated within the MyPAL application, meaning they can still play the quiz (e.g. Figure 2.2) and collaborative sorting game with or against the PAL actor. In addition, the MyPAL application provides a memory game, where instead of similar images, pictures of food and the number of carbohydrates that food contains are matched. The application also provides an interactive diary, where children can register their blood values, carbohydrate intake during meals and snacks, what they did during the day (e.g. school, sports) and how they feel (Ligthart, 2016). A subset of the interactive diary also allows the child to set the sentiment of the entry they are adding, by means of a slider. The picture shown in Figure 2.3 to the right of the slider changes along with the slider, to indicate the feeling the children have currently selected. Within the application children can also see the progress on their goals. Child interaction with the MyPAL application is stored on a central server, to provide insight to the HPC, parents and researchers into things such as how well the child is doing with their progress or feelings or how certain concepts influence each other.

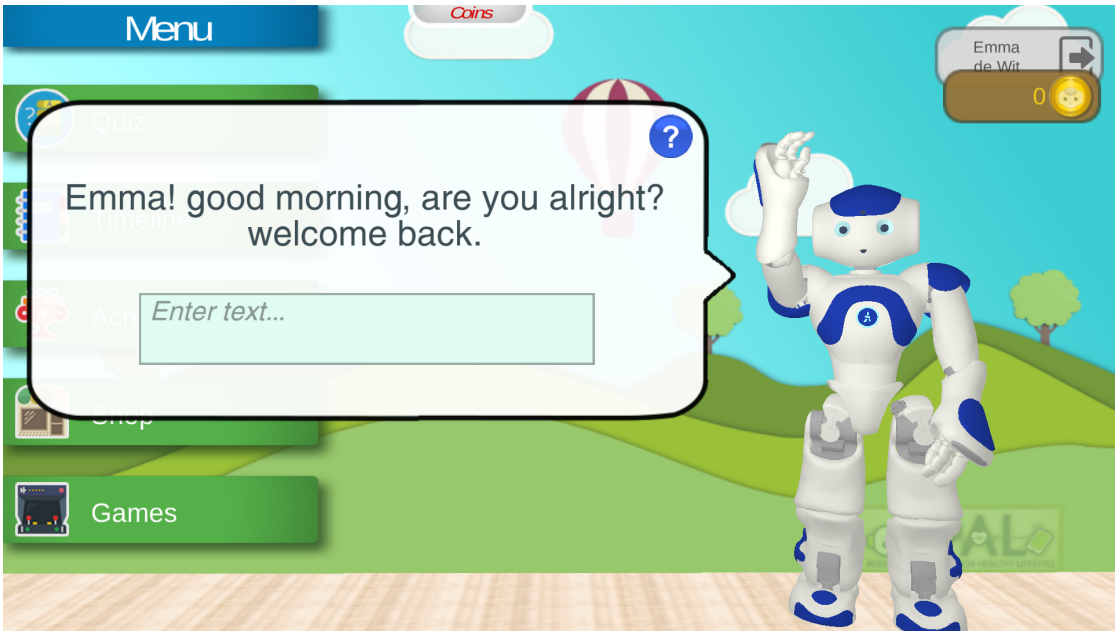


Figure 2.1: The robot's avatar greets child Emma after she logs in.

Figure 2.2: The child correctly (indicated by the green colour) answers a question asked to her by the robot's avatar. The child is up next to answer a question, which the robot will answer.

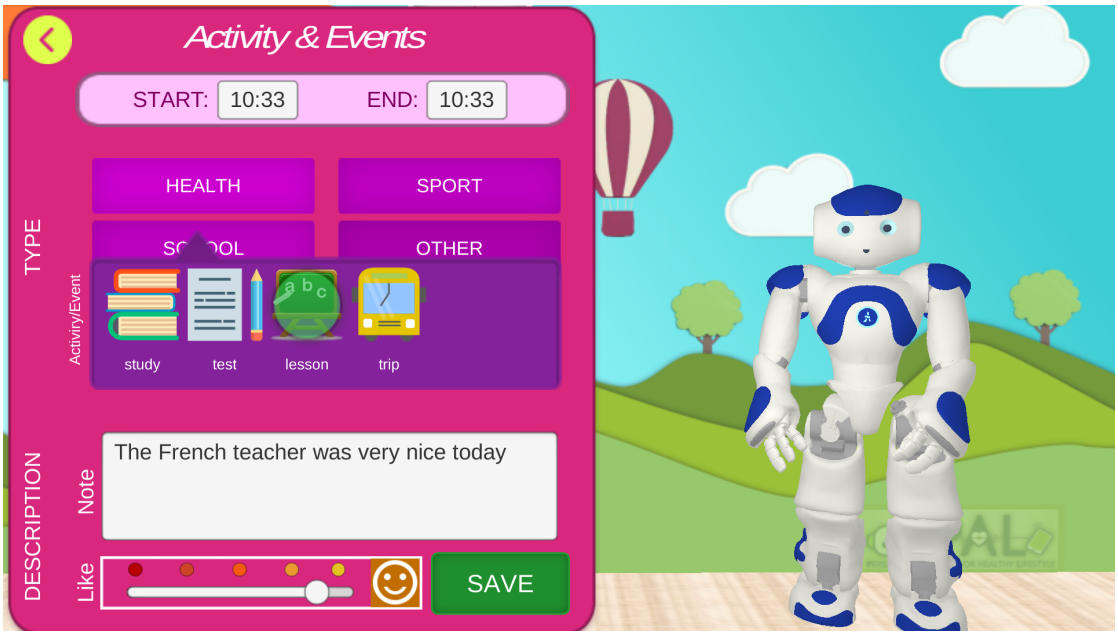


Figure 2.3: A screen shot of the interactive diary within the MyPAL application, where children can add meal information, blood glucose measurements and activities of their day. Children can convey how they feel about a new activity by means of an emotion-slider.

2.1.4 The PAL cloud

The entire PAL system is driven through a central server. This ensures the behavior of the physical robot in the hospital and that of its virtual avatar in the MyPAL application do not differ from the perspective of the child, ensuring that the avatar is a believable representation of the robot itself.

Apart from the behavior of the robot, the central server also provides the means to a HPC to change a child's goals through the web application *PAL Control*. All available goals are provided there in a tree view (Peters et al., 2017) and can be selected or deselected by the HPC. This web application also allows the HPC to monitor progress of the child. Figure 2.4 shows an overview of which systems communicate with the central server (indicated as the *PAL Cloud*) and when those systems are used by which users. Note that the *PAL Monitor & Inform* is not currently implemented.

The central PAL server consists of many modules, each of which is responsible for a specific set of behavior or logic of the PAL actor. All modules, as well as the MyPAL application, send instructions and information messages through a central channel, which each of the other modules can react to. For example, the *Goal Calculator* module calculates which task (e.g. which specific quiz question) to provide to the application if the child initiates an activity (e.g. the quiz), so that that task is relevant to the current goal and has not yet been achieved. A sentiment miner processes all open text entered by the child and assigns a sentiment value to

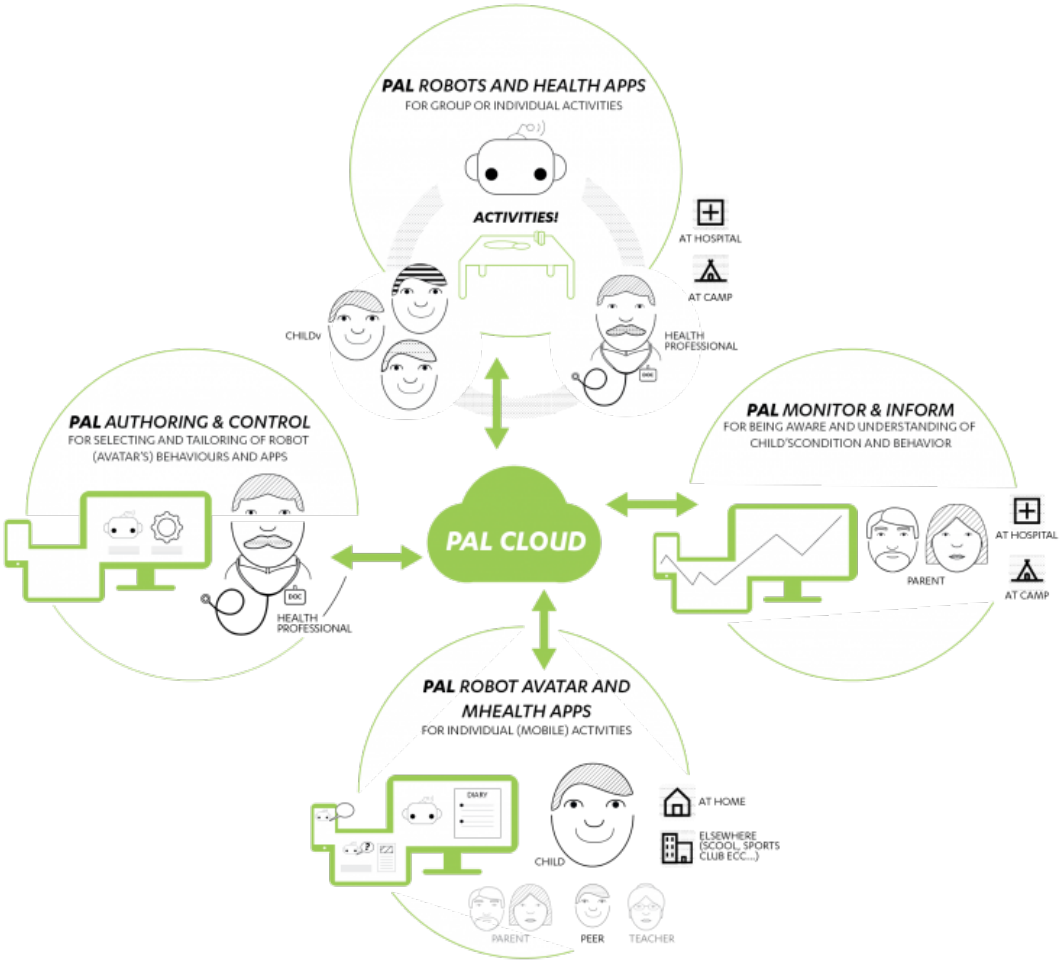


Figure 2.4: The PAL Cloud, taken from http://www.pal4u.eu/index.php/project/about/, see also Ligthart (2016)

that string (positive +, negative − or neutral 0). The episodic memory manager by Schreuder Goedheijt (2017), which was mentioned in the introduction, is also a module that works in this system, and is discussed in more detail in Section 2.5.3 on page 26.

All the systems knowledge is made persistent in an ontology based database, which will be discussed in Section 2.2 on the next page.

The behavior and dialog managers send instructions to the robot (if in use), and to the avatar in the application to control their movement and speech. Speech is generated through so called action plans in the *CPlan* language (Kiefer, 2012). This language allows defining a grammar, where a plan is matched against a current situation. If a grammar statement matches, the sentence can be generated using a script, information from the database that is true at that time can be used in the final dialog act that is used, and randomization can vary the words that are

used at a specific place in the sentence, to increase variation in the PAL actor's utterances.

2.2 Ontologies

An ontology is a directed graph that is used to represent real-world objects, and to specify properties of and relations between those objects – often in a specific domain. Classes may have individuals, which are instantiations of those classes: A class describes the object, an individual corresponds to a unique instantiation of it. The concept of ontologies is a formalization of how set-theory and predicate logic can be used to describe and interrelate real world concepts and are useful to encode domain knowledge in a format that can be used by artificial agents.

At its core, an ontology is a collection of *triples*. A triple is a combination of *subject* - *predicate* - *object* and defines what properties a class has, or how one class is related to another. The best way to explain how such triples could make up an ontology in practice is by using an example, and a widely used example of how to use ontologies is the concept of Pizza's (Rector et al., 2004). For the rest of this section, the `Pizza` ontology will be used to exemplify ontologies. In this example, a single triple could look like this:

```
Pizza hasTopping Topping
```

which expresses that the class `Pizza` has a relation with the class `Topping`, and that relation is called `hasTopping`.

In reality, these triples do not just consist of words, but of URI's (Unified Resource Identifier). The class `Pizza` needs to be defined in some ontology, which is not necessarily the one where the triple is defined in. If the classes `Pizza` and `Topping` are defined in the Stanford Pizza Ontology[4], the same triple could then be expressed using *prefixes*:

```
https://protege.stanford.edu/ontologies/pizza/pizza.owl#Pizza
https://protege.stanford.edu/ontologies/pizza/pizza.owl#hasTopping
https://protege.stanford.edu/ontologies/pizza/pizza.owl#Topping
```

Since it is likely multiple classes will be used for the same ontology, an alias is often defined for the complete URI – the namespace – of an ontology, so that if the alias for `https://protege.stanford.edu/ontologies/pizza/pizza.owl` would be `pizza`, the entire triple could be expressed like this:

[4] `https://protege.stanford.edu/ontologies/pizza/pizza.owl`

`pizza:Pizza pizza:hasTopping pizza:Topping`

By using URI's and namespaces, classes and relations that have already been defined previously can be imported in a new ontology and thus be reused.

An extension to the core concept of ontologies is given by the Resource Description Framework (RDF, Klyne and Carroll, 2006). RDF adds some extra functionality to the basic triple format that allows describing how certain classes interrelate and what the domain and range of a predicate is. For example, using RDF, an individual can be said to be an instantiation of a class using the relation `rdf:type`, or a class can be made a subclass of another[5].

On top of RDF, various other (competing) extensions are built. A common one, and the one used for the PAL project, is called *OWL* which stands for Web Ontology Language (Bechhofer, 2009). OWL adds reasoning options to the ontology, such as specifying formal logical relations such as transitivity or symmetry. This allows building an ontology with implicit facts, possibly decreasing the number of explicit facts necessary for a complete ontology. It also provides the possibility to specify two classes as being equal[6] or distinct.

2.2.1 HFC Database

Ontologies can not only be used as knowledge graphs, but as dynamic databases as well. This means information can selectively be retrieved from the ontology, or new facts can be added, using queries. Where traditional relational databases consist of tables, where each row is a single entry, information in an ontology database can be interrelated with every other part of the ontology. That means there is no easy way of identifying single 'entries' in an ontology. Instead of a database scheme that lays out what tables there are and how they look, an ontology defines what classes exist, what their properties are and how they relate to each other. The data in the ontology database is then made up by individuals, which can be added, removed and queried while the database is in use. Individuals may be specified in the ontology describing the database itself, but often this ontology only specifies what the data should look like, not what it is.

In ontologies, information is requested or added in the form of triples, to reflect the underlying data. It is not, like in traditional databases, specified by the location in a specific table. For ontologies such as OWL, the query language is called

[5]This is done with the predicate `rdfs:subClassOf`. Note that RDFS stands for Resource Description Framework Schema, and belongs to RDF.

[6]Technically, $A \equiv B \iff A \subseteq B \wedge B \subseteq A$, so the same effect could be achieved using the `rdfs:subClassOf` predicate. However, OWL also also adds the possibility to specify such things as version and compatibility information, making the OWL semantics more fit for importing external ontologies

SPARQL. A very simple query, to select all individuals of the class `Pizza`, would look like this:

```
SELECT ?pizza WHERE ?pizza <rdf:type> <pizza:Pizza>
```

Variables in a query are prefixed with a question mark.

The PAL Project makes use of a database that is built on top of OWL – so ontology editors such as Stanford's Protegé can still be used, but which extends the triples with a fourth value: the transaction time, which is the exact time a triple is added to the database. This extension is called the *HFC Database* (Krieger, 2012, 2016) and the HFC query language is similarly an extension of a proper subset of SPARQL.

The advantage of HFC is that it allows facts to change over time, without having to remove the asserted triple, as would be the case in traditional ontology databases. The previous query in HFC would like like this:

```
SELECT ?pizza ?time WHERE ?pizza <rdf:type> <pizza:Pizza> ?time
```

This query would not only return a list of individuals that are pizza's, but also an 8 bit UNIX time stamp indicating when the triple specifying that individual to be of the type pizza was added to the database.

Usually, ontology databases operate under the closed world assumption, that is to say, if a fact is not stored in the database, it is assumed not to be true. This means a triple should be deleted from the database if its truth status changes over time. The consequence of this is loss of information, which is why the HFC database is used within the PAL project.

With the HFC database, a fact is assumed to be true starting from the moment it is added to the database. Once the truth value of the triple expires, the original triple can be inserted again, with the explicit polarity that it is false. Because all triples have a transaction time associated by default, not only can the database be queried for what is true right now, but also about what has been true in the past. This allows researchers and HPC's to gather data about the interaction of the child with the system over the duration of the entire session.

The entire ontology used in the PAL project is a relatively large project. For structure, the ontology is split into seven separate ontologies, that are all joined by an eighth top level ontology, which is called the `PAL` ontology. The other seven ontologies can work completely agnostic of each other and can be ported to a different project without having to redefine domain specific values or relations. The seven ontologies, along with a very limited subset of classes, is visualized in Figure 2.5 on the following page. The ontology called *Upper* defines some abstract concepts of entities. Using this ontology, entities can belong to one of the three

disjoint classes *Abstract*, *Physical* or *Happening*, each of which have there own set of properties useful for the specific class of entities. Another ontology used is *Time*. Here, two properties for classes are defined, namely *diachronic* and *synchronic*, which indicate the property cannot or can change over time respectively. Where the HFC extension allows properties to change over time, this specific ontology can be used to specify which values are allowed to do so. For instance, the home address of a child can change (so this property is synchronic), while the date of birth cannot (making the property diachronic).

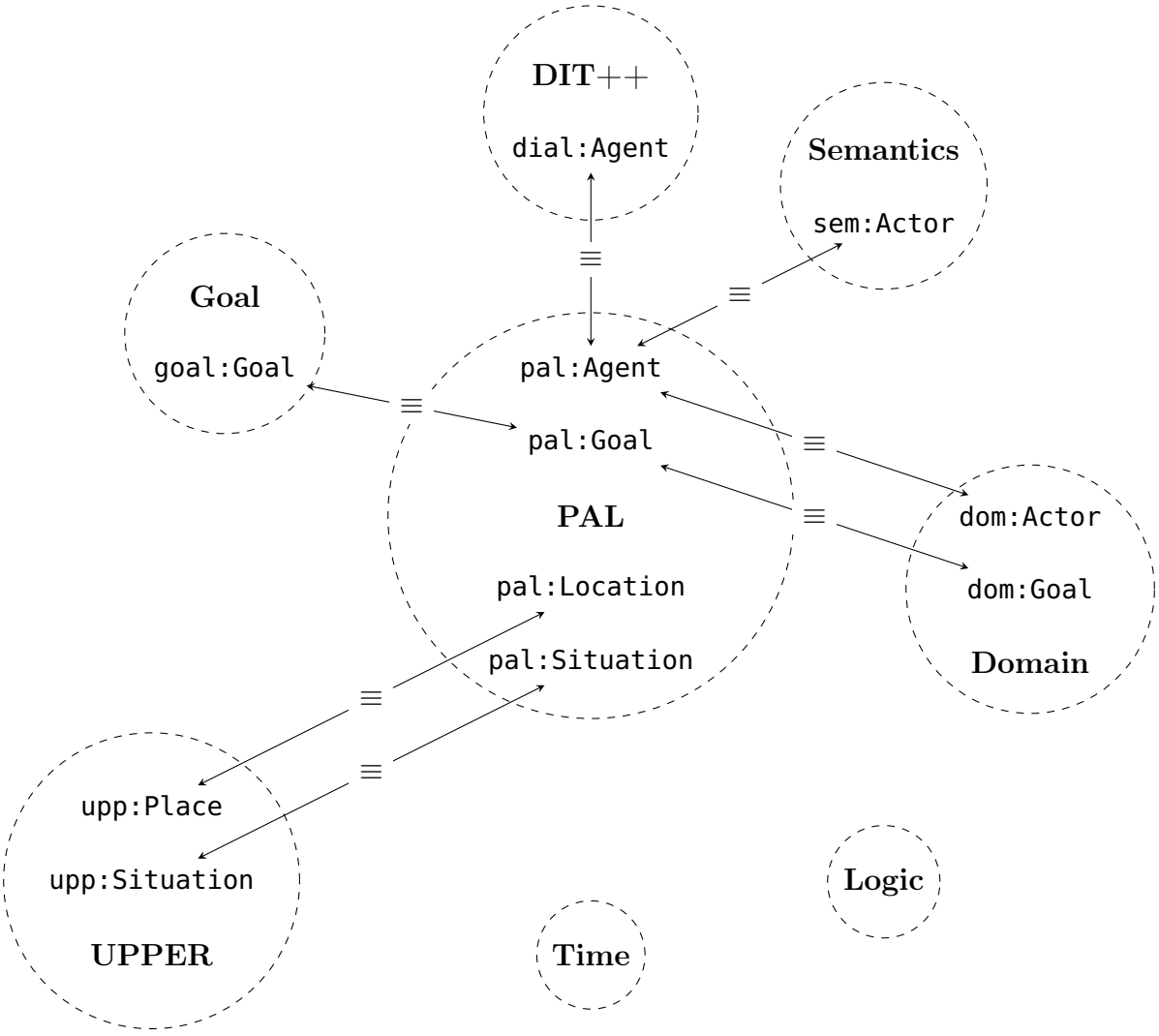


Figure 2.5: The central PAL ontology connects classes and properties of the other 7 ontologies using relations such as equivalence.

Domain specific knowledge is described in the *Domain* ontology. This is where concepts such as `Diabetes`, `MeasurementDevice`, `Actor` – which can be a participating child, the PAL actor, or family of the child – are defined. This ontology is designed specifically for the diabetes oriented PAL project and would not, contrary to the other ontologies, be ported easily to another domain without rigorous

refactoring.

For a more complete overview of the ontologies used within the PAL project, and their underlying relations, see Krieger et al. (2016).

2.3 Affect

In the context of the PAL project, an ontology for storing and reasoning over affect was developed by Sternheim (2017). From the start of the PAL project, a single class `Mood` was included in the `Domain` ontology as a synchronic property of the child, and could take values such as *happy*, *depressive*, *apathetic*, *angry*, *traumatic*, etc. The initial intent was that the HPC would estimate the mood of the child during intake and set it in the system, which would then update it where necessary. The child's mood would subsequently be used to adapt the PAL actor's dialog to be more empathetic or *affective*. Throughout the duration of the PAL project however, this value was always instantiated to *happy* by default, and did not change over time, as no clear idea existed of what the concept *mood* meant, or how to work with it.

2.3.1 Emotional State Model for Affective Semantics

Sternheim developed a more precise way to model the emotional state of an agent in her ESMAS (Emotional State Model for Affective Semantics) and created an ontology to represent that model.

Mood and emotion are two different – although closely related and interacting – concepts that together form the *emotional state* of an agent. The umbrella term *affect* is often used to refer to any of these concepts alone or together, but is less specific and should not be confused with any three of them. The mood of an agent is a relatively long-lasting feeling, which is often experienced with less rigor than emotion. It is always present, while emotion is always evoked by a direct stimulus, and may thus be absent if no such stimulus occurred recently. Emotion is how we experience events that happen in the world. The emotion that is elicited by a certain stimulus depends not only on the stimulus itself, but also on the agent's mood at the time of that stimulus. A positive mood will positively affect the elicited emotion for a certain stimulus, as will a negative mood negatively affect it. Mood, on the other hand, is also influenced by emotion, be it less strongly. For example, a positive emotion will slightly shift the active mood in the positive direction as well.

Both mood and emotion can be expressed in two dimensions: *valence* and *intensity*. The valence is a metric for how positive or negative an emotion or the

mood is. For example, 'happy' has a high valence and 'sad' has a low valence. Intensity indicates how intense the emotion or mood is felt. For example, 'happy' and 'ecstatic' may have similar valence, but 'ecstatic' will have a much higher intensity. Since mood is experienced with less vigor than emotions are, they typically have a lower intensity.

Intensity and valence are not fixed, absolute values, but constitute ranges. Using such ranges, instead of merely the associated labels, allows to encode a wider variety of moods and emotions, since some specific feelings may not be directly expressible using a single label. Further more, not all labels correspond to the exact same sentiment across languages. For example, the Dutch word '*blij*' partly covers the ranges of both English words '*glad*' and '*happy*' although sometimes neither word would be a correct translation. Therefor, using ranges instead of labels could offer a better way to encode affect in multilingual systems such as in the PAL project.

A visualisation of the ESMAS, which models the interaction between mood and emotion given valence and intensity values, is given in Figure 2.6. An ontology has been created of this model, that allows the emotional state of an actor to be represented in terms of mood and emotion, both described by a label, or its valence and intensity values. Sternheim has used a classification of emotions from the literature and annotated these emotions with their corresponding labels and valence and intensity values. These were then used to instantiate emotions as individuals in the ontology, which enables the system to infer an emotion label from the intensity and valence values and vice versa, using a single query. This means either representation can be used to encode a given emotion and the other is automatically inferable by the system. A similar effort has been made for mood. However, since mood is harder to describe and, due to its long lastingness, harder to elicit in an experimental setting, the literature is less clear on classes of moods. In the ESMAS ontology, two classes are present: positive and negative mood, both of which have low intensity and a positive and negative valence respectively.

Emotional states are given by four classes in the ontology, given by the polarity of the mood and emotion at the time of the emotional state. Using these four classes, the PAL actor's dialog could be matched to the child's emotional state. For example, when the child experiences a negative emotion but their overall mood is positive, Charlie could utilize the knowledge that moods are longer lasting and say the child will likely feel better soon, based on the knowledge the underlying positive mood lasts longer than the current emotion.

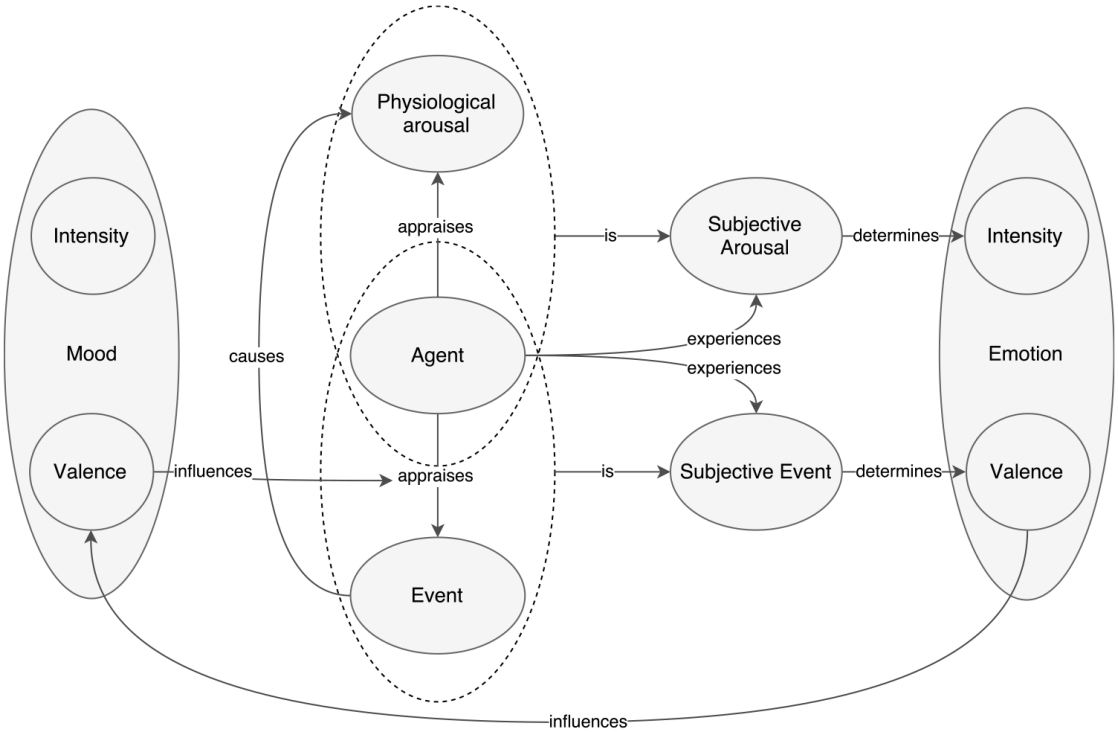


Figure 2.6: The ESMAS model (Sternheim, 2017, p.13) models how mood and emotion influence each other.

2.3.2 Sentiment

Recently, efforts have been made to extract sentiment from child utterances in the MyPAL application. This analysis can work on utterances outside the diary component, where children can already indicate sentiment manually using a slider (see Section 2.1.3 on page 7 and Figure 2.3 on page 9).

Sentiment analysis may provide a good starting point to find the emotion and mood of a child, and trace these over time. However, while the ESMAS describes in detail how mood and emotion interact, sentiment is as of yet left out of this model. The sentiment slider gives a sentiment value on a one-dimensional 5-point scale, rather than in terms of valence and intensity. The sentiment mining tool is even less precise and returns only one of three possible sentiments: positive, negative or neutral. Although a few possible mappings between sentiment values and valence and intensity could be conceived, there is no clear indication as to what mapping would be correct. Sternheim leaves this as an open question.

Hovy (2015) provides an analysis of the classical interpretation of sentiment, and how it relates to the concept of emotion, but also points out some limitations of sentiment analysis. He indicates sentiment constitutes an *attitude*, and that attitude must be directed at something. This is different from mood, which is always present, and emotion, which is caused by a stimulus, but not necessarily

directed at something.

One way of sentiment extraction, similar to what is used in the PAL project, is to look at certain key words which indicate either a positive or a negative sentiment. If neither prevails, the neutral sentiment will be attributed to the utterance. However, since sentiment is directed at something, that *something* may be a very specific thing. Consider a hypothetical sentence a child may say to the Charlie:

> "I went to school today and gymnastics was a lot of fun, but I didn't like the book we had to read"

There are three possible topics the sentiment may be directed at: school, gymnastics, and a certain book. While it may be clear that the hypothetical child attributes a positive sentiment to gymnastics class and a negative sentiment to the book that was read, it is not immediately clear what the child thinks about their day at school or, indeed, what the sentiment of this utterance is.

Hovy proposes not only to look at the raw sentiment data, but at the underlying motives for that sentiment. Peoples mood, emotion, but also internal goals may explain *why* certain sentiments are held.

2.4 Question asking

In day to day life, question asking makes up an important part of human conversation. Kearsley (1976) was one of the first to provide a taxonomy of the types of questions we use (see Figure 2.7), and also provided an overview of *how* we use these different types of questions.

2.4.1 Question types

Kearsley first divides questions into three classes: *verbal*, *non-verbal* and *indirect* questions.

Verbal questions are those that are clearly identifiable as questions and, in written text, are ended with a question mark. Indirect questions are usually statements that elicit further information or invite to confirm or reject what is being said. These type of questions usually do not have a clear question mark or inflection rise at the end of the sentence. Non-verbal questions are not spoken aloud and can be either *overt* or *covert*. Overt non-verbal questions are conveyed by means of facial expression or gestures, while covert questions that are not directed at other people, but at ourselves.

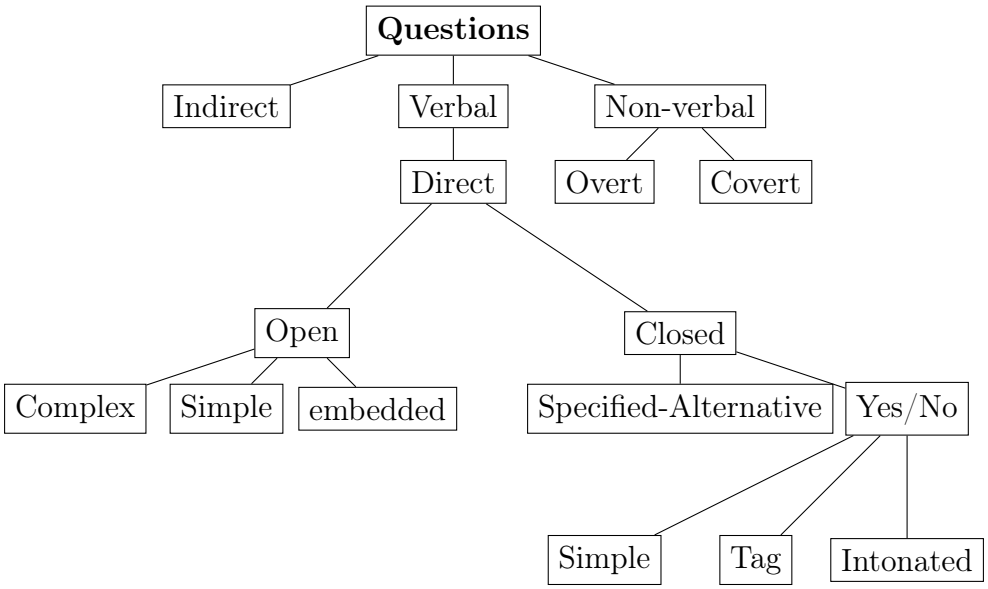


Figure 2.7: A taxonomy of question forms (Kearsley, 1976, p.357).

For the PAL project, the verbal questions are of most interest, since, according to Kearsley, these questions are always direct. These direct verbal questions can be either *open* or *closed*, where closed questions can be *yes/no* questions or *specified-alternative*. A yes/no question is a question where the answer is always yes or no, while a specified alternative question embeds the possible answers in the question ("is that animal a cat or a dog?"). Open questions are those questions that allow a wide range of answers and are often used when the number of possible answers is too large to specify. These questions are often also referred to as 'wh-questions', referring to their usage of the who/what/why/where/when/how question words and are typically answered with a *wh-inform* statement. Open questions can be further classified as either *simple*, *complex* or *embedded*, where simple questions contain only a single wh-word (*"who is that?"*), complex questions contain at least two wh-words ("who said what to whom?") and embedded questions, which are questions embedded in another question or in a statement ("can you tell me what time it is?").

The ratio open to closed questions that is asked in a conversation depends on the social setting of that conversation. In formal settings, more open questions are asked, but when both conversationalists know each other well, more yes/no questions are asked. In these settings, those questions are rarely ever answered with a simple yes or no, however, and usually elicit a more extensive response (Kearsley, 1976, pp.370-372). Further more, questions not only *depend* on the social status, but can be used to *set* the power structure as well. Asking a question directly after the conversation partner gives an answer without allowing them to ask questions, is often used to exert power, while using embedded open questions indicates a more equal power structure (Kearsley, 1976, pp.366-367).

2.4.2 Question functions

The function of questions is to elicit response in a conversation partner, but the intended response depends on how the question is asked. Kearsley provides a taxonomy of question functions (see Figure 2.8). An *echoic question* is asked to verify the interpretation of something that was said by the conversation partner. An echoic question can be "Huh?" or "What?", but can also be used to clarify something specific. For example, when a child tells Charlie they went to a birthday party the day before, but the parse certainty of the conveyed time is too low, Charlie could ask the echoic question "*When* did you go to a birthday party?"

Epistemic questions are specifically intended to request more information from the person the question is directed at. Epistemic questions can be either *referential* or *evaluative*. The former are questions intended to specify specific knowledge about a certain context. This can be, for example, informing who the specific person the conversation partner talks about is, asking to be more precise about the time something happened, requesting a certain procedure to be explained in more detail, etc.

Evaluative questions are intended to test the listeners knowledge on a specific subject. These are the type of questions that are used in the quiz game within the PAL project.

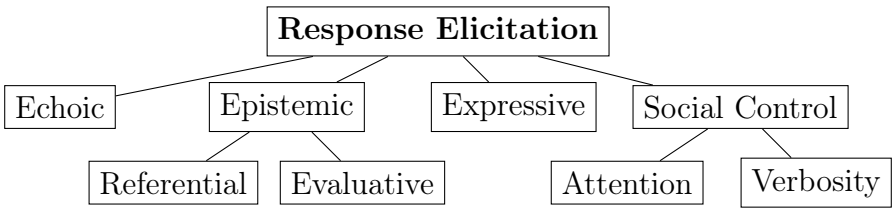


Figure 2.8: Functions of questions (Kearsley, 1976, p.360)

Kearsley claims people have an internal knowledge base, and questions arise when that knowledge base needs to be updated. This can be when specific information is missing about a certain subject, or when confusion arises, for example when conflicting information is acquired. In case of the latter, the question is intended to figure out which of two pieces of conflicting information is correct. Questions can be formulated to enforce a believe which is more likely, or reject a belief that is less likely. Questions can also be used to fill in gaps, using one or more of six frames of reference: space, time, properties, causes, procedures and roles (Kearsley, 1976, p.364).

2.4.3 Question asking in listening agents

The approach of filling in the gaps to form questions was used by Han et al. (2013) to create a counseling agent. During a conversation between the counseling agent and a user, each user utterance is broken up in the 5W1H parts (who, what, where, when, why, how) and the sentiment, resulting in 7 possible *slots*. Since not each user utterance contains information on all these slots, questions can be asked to fill in the remaining slots. Their approach allows the robot to ask direct questions relevant to the information expressed by the user, thus expressing interest in what the user says.

Building on this approach, they developed a more advanced listening agent, which extracts the user intention from the utterance and selects a response appropriate for that intention (Han et al., 2015). Instead of 5W1H information, triple relations are extracted from the user's utterances. These triples have exactly the same meaning as in ontologies as explained in Section 2.2 on page 11, but are now directly extracted from natural language, rather than predefined. The triples serve to store relations between entities within these utterances. For example, when a user says "I like Messi," the objects 'I' and 'Messi' are connected through the relation 'like' in the triple `I - Like - Messi`.

After the triple extraction has taken place, named entity recognition is performed. This finds an URI in an external knowledge base for a named entity within the text. Using this external knowledge base, the notable type (in the case of Messi, the notable type is "Footballer") is found. A follow-up question is consequently formed in one of two ways. Either a question is asked about a related entity by finding the 10 most popular instances of the notable type, or a question is asked about a property of the named entity itself. This approach results in questions such as "What other football players do you like?" (using the property "is football player" of the named entity) or "Do you also like Suarez?" (using another instance of the notable type of the named entity).

Bang et al. (2015) go one step further. Instead of asking questions relating to the named entity directly, they store it as a topic of interest for later usage. Over time, this results in a database of topics the user likes to talk about. If, during a later conversation, the user requests a new topic, or a natural moment arises to switch the topic of conversation, an interest of the user can be selected from the database by finding topics which have been referred to in previous conversations. Bang et al. still make use of external knowledge bases for information on properties and notable types of the topics of interest. They not only use this information to formulate a question that relates to the topic of interest, but also to extract topics of interest that are never specifically mentioned. As an example, they note that a user may talk about the football player Messi, the football club Barcelona and the football league "La Liga" without ever explicitly mentioning football. Their ap-

proach can reason over common relationships and recognize football as the overall topic of interest of the user, which can then be used in future conversations.

2.5 Episodic Memory

Episodic memory is the memory system responsible for capturing specific events, or *episodes*. It is an *explicit* memory system, meaning the recalling of memories happens consciously, which differs from e.g. remembering how to walk, and is part of the *long term memory*.

Episodic memory allows us to relive a previous experience with as much detail as if that event is occurring right now. The term *episodic memory* was introduced by Endel Tulving (1972) to contrast classical views of human memory with the back then recently introduced *semantic memory*.

Semantic memory provided a new theory on how the human brain can have *general* world knowledge, such as knowing where Japan is on a map, without remembering the moment we learned that fact, as well as the specific memories of events. Before the introduction if this semantic memory, when human memory was studied, human participants were tested for their ability to remember specific things that they experienced, encountered or studied previously.

Although Tulving first used the term '*episodic memory*' to refer to any type of memory that is not semantic memory, it later turned out episodic memory differed from the classical views of memory as well. The initial studies of memory concerned *what* was remembered, while episodic memory also concerns *when* and *where* that event took place. Later, Tulving revisited the theory of episodic memory and detailed everything that had been learned about it by then (Tulving, 2002). In that essay, he listed three important properties of episodic memory, that made it different form all other types: The sense of the self, subjective time, and autonoetic awareness.

The sense of self, sometimes also referred to as *autobiographic memory*, means that ones memories need a perspective and this perspective is always that of the rememberer. Subjective time, as Tulving used it, is the concept that the human mind is able to mentally travel back in time, and re-experience facts that are no longer happening. This is the greatest distinction from semantic memory, as it makes the memory not just a fact we know, but places it in the context of the time and place of the original event. Lastly, autonoetic awareness is the awareness that these episodic memories are mental experiences and, although *experienced* as if they are happening at the time of recall, are not *actually* happening at that time. Autonoetic awareness allows us not only to mentally travel back in time to previously experienced events, but also to imagine – and thus reason about –

future events or places where we are not right now or may even have never been, without ever confusing those imaginations with reality.

In addition to these three properties, several more have been suggested in the literature. Jockel et al. (2008) mention that episodic memories are *imperfect*. Memories in this system are subject to change: when we relive an episode, the episode in our memory is changed by how we re-experience it and the different emotions we feel at the time of recall. Furthermore, episodic memories are easily forgotten. This happens when a memory is not recalled for a longer period of time, and when the emotional valance is not high enough, in other words, it did not make enough of an impression. Forgetting is not a discreet process, but rather, the probability of recall decreases over time and only increases again if it has been recalled. This process is called *memory decay*. Lastly, episodic memories are *temporally indexed*. Even when we cannot pinpoint exactly when a certain event took place, we can usually say when it occurred relative to other memories related to the subject.

2.5.1 Episodic Memory in Cognitive Architectures

The theories underlying episodic memory have attracted some interest from the field of intelligent agents, as reliving previous episodes can help predict the effect of certain actions in an environment, allowing more effective action formulation in novel environments and situations. Perhaps the first who noticed this potential were researchers of cognitive architectures. A cognitive architecture aims to create a unified implementation of various processes in the human brain to verify the *plausibility* of theories regarding these processes. Nuxoll and Laird (2004, 2007) implemented a model of episodic memory, called `EPMEM` into the cognitive architecture `SOAR`. They focus on encoding, retrieval, and event segmentation. The latter is solved by segmenting periods of time into episodes when a subgoal of the agent changes. When this happens, the entirety of the working memory is written to the episodic memory as a snapshot. This snapshot contains information about the intended goal, the actions that were taken and the actual results of those actions. This deals with the encoding of the episodic memory, although this process was refined in Nuxoll and Laird (2012), where the encoded working memory is transformed to a graph structure of triples (identifier - attribute - value). Retrieval is done by matching current goals with memories of how similar goals were achieved previously. These memories are then used to predict the outcome of a certain set of actions and compare that predicted outcome with the current goal.

The `EPMEM` module has since become an established part of the SOAR architecture. Gorski and Laird (2011) used reinforcement learning to enhance the recall process in order to increase the significance of retrieved memories. Later, the same module was used by Anderson (2015) not only to learn which actions to take in

various situations, but how to explain the actions the agents finally settled on automatically as well. Brom et al. (2010) went for a different approach and used the `EPMEM` module in `SOAR` to test the effectiveness of intelligent agents enhanced with episodic memory in more realistic world scenario's. Their main research focus was believability, rather than plausibility, since they hope to utilize their results for non-playable characters in video games.

2.5.2 Episodic Memory in Social Robots

For social robots, memory may fulfill a different role than action selection in novel environments, which is the main interest of the current research: for a social robot to interact with a human user in a meaningful manner over long time, the robot must be *believable* in its social interaction. This had already been recognized in the PAL project's precursor ALIZ-E:

> "In attempting to achieve long-term social interaction, there is a clear necessity for memory to link prior experiences with ongoing and future episodes" (Belpaeme et al., 2012, p.39)

Castellano et al. (2008) identify three important factors for long-term social interaction: affect sensitivity, memory and learning, and cognitive and expressive behavior. Affect sensitivity means a social robot should be aware of the affective state of a user, be able to respond accordingly, and, where appropriate, effect a more positive state. Memory and learning refers to the required ability of a social robot to remember past experiences of interaction with the human user. They stipulate the challenge is not only how to encode memories and the affective state of the human user, but also to recall such memories at the appropriate time. Lastly, the social robot should exhibit cognitive and expressive behavior, which enhances the believability of the robot if done properly. This component should be well integrated with the memory and affect.

One approach to social robots with memory is that of Ho et al. (2009). They created a social software agent which can switch between a physical and digital body and environment. The agent uses its episodic memory to maintain a clear sense of its own identity across those transitions and learn not only about its physical (or digital) environment, but its social environment as well. This means the agent can learn not only how certain actions affect the environment, but what the activities of the user it supports are as well. The agent also possesses an internal emotional state, to which all memories are related. Episodes without emotional relevance are discarded over time, so only relevant episodes remain. All this allows the agent to adapt to its human user over time, regardless of the body and environment it is inhabiting.

Kasap and Magnenat-Thalmann (2010) developed a social tutor robot called Eva, who tutors her students in the subject of computer networks over the course of two sessions and exhibits supportive behavior. Eva has goals (e.g. teach the user) and beliefs (e.g. the user is friendly or the user is capable). Memories are encoded using context (the initial state of the robot and the desired outcome), contents (a specification of what happened during the memory event), and actual outcome. By asking direct questions, Eva can learn things, such as the emotional state or level of motivation, from the user. This is used in the second and last sessions to personalize the lesson. In this session, Eva can also refer back to goals she had during the previous session and whether those goals where achieved, allowing her and the student to pick up where they left of.

The aim of this approach is to exhibit intelligence, which should help keep the user's attention for a longer time. According to subjects, the use of episodic memory increased the social presence of the robot. Episodic memory also resulted in a higher task engagement of participants and the novelty effect did not wear of over the course of two sessions. However, participants were not significantly more motivated if the robot was supportive than if it was not (Kasap and Magnenat-Thalmann, 2012).

An approach that is perhaps closest to what is intended for the PAL project is that of de Campos (2010). He developed an episodic memory system, called MAY (my Memories Are Yours), which is based around a structured ontology and uses the concept of *shared memories* that was also used by Schreuder Goedheijt (2017): memories of the human user which they voluntarily share with the system.

The MAY system is a long term robot *companion*, which should support the user over a large part of their life. MAY uses the shared memories to learn about the (life) goals of the user, but also about patterns of events in the users life, which allows it to reason over possible future actions. The memory system that was designed is quite extensive and consists of three levels of abstraction, based on literature from the field of psychology.

The highest level consists of *Life Time Events*. These are long important periods in a users life, such as *childhood*, *primary school*, *first job*, *19 years old* or *bachelor*. Life time events are not disjoint, e.g. primary school usually overlaps with childhood.

The second level encodes *General Events*. These are specific occurrences of events, although these events may also last for longer periods of time. For example, a specific event is a birthday party a user may attend, but it could also be a 3-week holiday, or even the first year on a new job.

The lowest level consists of *Event Specific Knowledge*. This level encodes much shorter events, which can last from seconds to hours. In human memory these correspond to images of specific situations popping in one's head and is closest

related to how episodic memory has been described so far in this research.

All these three levels are interrelated. For example, the general event *birthday party* happens during the life-time period *primary school* and *childhood*, and contains the event specific knowledge of the birthday boy blowing out the candles.

2.5.3 PAL's Episodic Memory Manager

The Episodic Memory Manager (EMM) used within the PAL project (Schreuder Goedheijt, 2017) is used to add a level of empathy to the conversations between the child and the PAL actor. The EMM allows to ask meaningful questions about events the child has shared with the system through the timeline interface. When a child enters a new event, a new episode is generated. The structure of the episode is defined in an ontology. The main object in this ontology is the `memory`, which is based on the 5W1H (When, Where, Who, Why, What, How) principle. This structure is visualized in Figure 2.9. The 'when' is encoded through a subclass of the general `happening` class. This `happening` contains a start time and end time of the event. The 'where' contains a reference to the location the event took place in, while the 'who' is a reference to the child who experienced and shared the event. The 'why' is encoded as an episode tag, which is a static entity within the EMM ontology and refers to specific events, such as `GoalWasToDifficult` (the last time a quiz with goal x was played, a lot of questions were answered incorrectly) or `NotFeelingWell` (the child indicated she was ill, feeling nauseous, etc). According to Schreuder Goedheijt, the episode tag *contains* the 'what', but *implies* the 'why'. The episode tag is further associated with an *episode trigger*, which determines when the episode will be referred to by the PAL agent. In the current implementation, this can be on specific times during interaction, such as when the child logs in, when the child closes the timeline or when the child completes a quiz.

Episodes are constructed not when the events they capture takes place, but when they become relevant to refer to. For example, the `GoalWasTooDifficult` episode mentioned earlier is constructed when a child logs in and, during a *previous* session, a child had a lot of wrong answers during a quiz game. As soon as the episode is constructed, it can be triggered, after which it is sent to dialog manager. From there, the relevant information is combined with a predefined template where only a few variables (e.g. the name of the goal) can be adapted, which results in an utterance that is sent to the PAL actor.

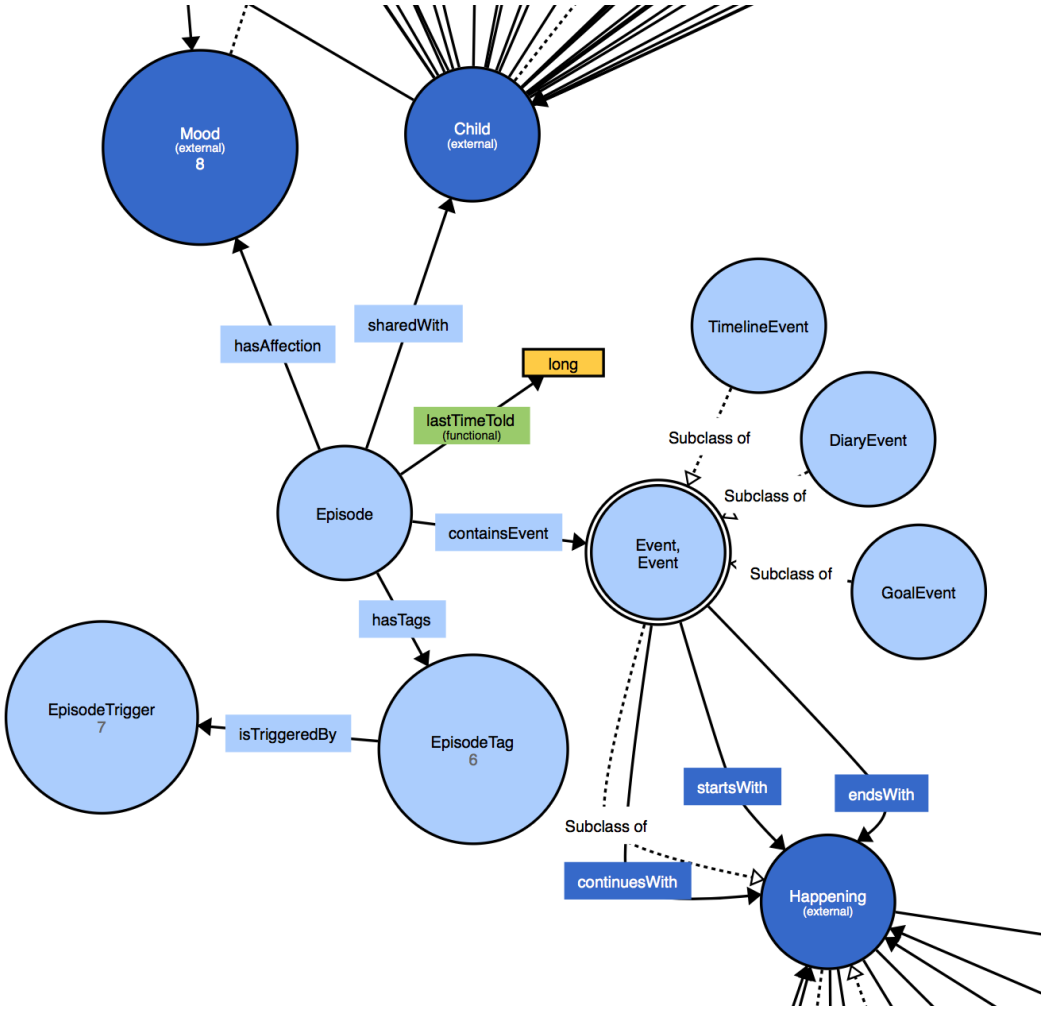


Figure 2.9: The core structure of the Episodic Memory ontology in the PAL project (Schreuder Goedheijt, 2017, p. 23)

Chapter 3
Design Specification

In this chapter, the minimal functionality of the ORACLE (Ontology Reasoner for Affective Conversation in Long-term Engagement) model will be specified. Next, a method for how the described literature will be used to achieve the design of this system will be described. Lastly, a method to evaluate the functionality on the basis of these specifications will be given.

At the start of this project, the final direction this project took was not yet foreseen. Initially, the aim of the current research was to extend the context of an episode describing shared memory that was used by the Episodic Memory Manager by making clever use of the functionalities already in place within the MyPAL application. Examples of such additional context was the exact text field a child was manipulating at the time the event occurred, how long such an event lasted, how often children login to the MyPAL application, etc. This initial approach is clearly reflected in the problem statement in Section 1.1 on page 1 and the description of the research questions in Section 1.3 on page 3.

During the first part of this research, it became clear this approach did not hold its own as well as was initially assumed and the literature provided far more promising directions that still – or even better – allow the answering of the research question and its subquestion. Due to these new insights, the focus was shifted from an integrated approach for personalized question asking on the basis of shared episodic memories, to an approach for encoding memories based on informative statements in natural language, and the reasoning over that encoding for question selection.

Although neither the original research question, nor any of the subquestions, changed with this shift of focus, the final approach places these questions in a new light.

The aim of this research, given direction by the literature, is to create a framework that allows affective question asking on the basis of the shared memories with a long term user. The ORACLE framework will be developed within the

context of the PAL project, but should be portable to other projects as well. This means that it should not depend too heavily on features that are available only within the PAL project, while maintaining compatibility with that system as it is now. The ORACLE should be a generalized implementation that is robust against future changes in the goals of the PAL project. If the list of required features is further extended, it should be sufficient to extend or adapt the ORACLE, rather than start again from scratch. With that in mind, potential future uses need to be taken into account during the design process.

Where possible, the ORACLE should be an extension of the Episodic Memory Manager (EMM) by Schreuder Goedheijt (2017). The current implementation of the EMM allows predefined interaction with the system to be translated to an episode with an associated meaningful question. The extension that is provided by the framework in this research should enable the PAL Actor to ask open questions on episodes from the child's life relayed to it by natural language.

Typically, the process for extracting episodic information from natural language and formulating a proper question could be divided into four parts: Content parsing, encoding, reasoning, and natural language generation. The content parsing and natural language generation parts will be assumed for this system. Instead, the focus will lay on the encoding and reasoning mechanisms, which will be encapsulated in the ORACLE model in this research, and implemented as an ontology.

3.1 General requirements

The purpose and goals described in Section 1.2 on page 2 and the research questions in Section 1.3 on page 3 can be made specific in the following implementation criteria:

1. The model should exhibit some form of memory

Question asking of the system is not a goal in itself. The main goal is to be able to relate previous episodes of the child to the current situation by means of questions by means of some sort of memory. Since this memory is related to *episodes* in the child's life, the type of memory the literature talks about is *episodic memory*.

2. The model should be able to encode informative statements.

The PAL actor should ask the child *referential epistemic questions* (see Section 2.4 on page 18). These are the types of questions used to infer more information from a conversation partner. The response to such a question is an informative statement, which conveys (new) information, typically in the form of a wh-statement. The system should be able to represent the information that is conveyed in the answers

to the questions it asks, as well as from informative statements put forward by the child on their own accord. Without this requirement, no context is available to the system to ask meaningful personalized questions.

3. The model should be able to encode and use the sentiment of the utterance.

Sentiment provides extra insight into the emotional state of the child at the time of their utterance. By encoding sentiment in relation to the utterance, the questions the system asks on the basis of those utterances can be further personalized, regardless of whether the question is asked right away or in a future session.

4. The model should be able to find topics for questions to ask.

This requirement gives context to the questions the system should ask. The system should (usually) ask open *wh-questions* questions. Both the topic to ask about and the type of question, indicated by the questioning word (who/what/where/when/why/how) should be determined by the model, although the actual formulation of the question in natural language is left to the dialog manager.

This requirement can be further split in the two following sub-criteria, which define how questions relate to encoded information, both from the current session and from past episodes:

4. (cont.)

   (a) The model should be able to infer missing information from similar episodes by generalization.

   (b) The model should be able to detect changes in patterns, and suggest questions on those changes.

5. The model should be able to reason over the affective state of the child.

Currently, the PAL system does not yet provide a way to determine the affective state of the child, or follow how it changes over time. However, the ESMAS does provides ways to encode this state. The ESMAS should be integrated in the PAL system, so that when in the future the affective state of children can be determined, the system can make use of the ESMAS to store and reason over that information. By making use of the affective state of children, questions can be personalized further than by usage of sentiment alone.

6. The model should store the unaltered input the children give to the PAL actor, and relate it to the episodes that are generated from that input.

Child generated text is useful for research, certainly within the context of the PAL project, since this can serve as a training corpus for natural language processing tools. For example, traditional sentiment mining works less well with utterances from children, as child speech is often less structured and may even use different or incorrect words. Storing the child utterances allows creating a corpus consisting of child-generated text that was uttered in a natural context, which could advance the state of the art in NLP related machine learning tools.

Furthermore, the child utterances could provide valuable insights in the emotional state and progress of the child that may not be readily available in the processed data (i.e. the encoded episodes). The unprocessed input may provide insights that help the HPC adjust the treatment or indicate situations to the parents they would not otherwise be aware of[1].

Lastly, by relating the unaltered input directly to the encoded episode that was generated from that input, errors in parsing, processing and mining can more easily be traced.

3.2 Episodic memory as analogy

Section 2.5 on page 22 contained a discussion of episodic memory, since this is the memory system after which Schreuder Goedheijt (2017) named his model for question asking based on shared memories with the child. As the literature shows, this is quite a common analogy for event related memory systems in artificial intelligence.

In the literature, a clear division between plausibility and believability could be seen. Plausible implementations try to simulate processes – such as episodic memory – according to theories on how these processes work in the human brain in order to test if those theories bring about the expected behavior and can therefore be said to be plausible. These types of implementations are often seen in cognitive architectures as described in Section 2.5.1 on page 23. On the other hand there are implementations with the goal of believability, where the goal is not to simulate the processes in the human brain faithfully, but to believably produce similar behavior. Examples of this were seen in the implementation of episodic memory in social robots or for non-playable characters in video games, as described in Section 2.5.2 on page 24.

The goal of this research is to be able to personalize the robot by using episodic memory to ask questions. For this, believability is of more importance than plausibility, which makes it useful to determine which factors from theories of episodic

[1]Privacy considerations should of course limit the amount of personal information of the child that is disclosed to the HPC and parents, see Fumagalli et al.

memory are necessary or useful for this system. This will be done by means of the list of properties of episodic memory given in Section 2.5 on page 22.

3.2.1 Autobiographical memory

According to theories from psychology, the perspective of an episodic memory is always that of the one that experienced the memory (Tulving, 2002). In the context of the PAL project, the PAL actor remembers – and reasons over – the memories of the child. Initially, this may suggest the memories are encoded from the perspective of the child – not that of the PAL actor. However, the memories stored in the system will not be of the events experienced by the children themselves, but of the child telling Charlie about those experiences. The conversation this happens in will be the true source of the memory, and that memory will be from the perspective of Charlie. It will only have memory of what it was told, not of the event itself. Thus considered, the autobiographic aspect of memories is useful to incorporate.

3.2.2 Subjective time

Subjective time allows a rememberer to mentally travel back in time to re-experience a remembered event. This is an important property for the PAL robot, as this allows the robot to compare current events to episodes in the memory. Relevant memories need to be readily available at any time and be placed within the context of that time.

3.2.3 Autonoesis

Autonoesis concerns the awareness of time, and of the fact that memories which are relived in subjective time are only imaginations and not the current reality. All the properties of episodic memory discussed here could be made subject to a more philosophical debate on whether computer programs could have these properties, but on the subject of autonoetic consciousness, this discussion becomes most prevalent. For autonoesis, consciousness seems to be a prerequisite. The debate on whether a sufficiently advanced artificial intelligence could posses consciousness – much less if the PAL system in its current form is sufficiently advanced – will not be discussed here. Since autonoesis is one of the three key properties of episodic memories listed by Tulving (2002), however, this property had to be included in this overview for completeness sake.

The main comparison that can be drawn to the PAL system is that the robot should not confuse episodes from the past with current activity. This is, however, a consequence of how memories are used by the system and not a property of the

memory model itself. Autonoesis is therefore not a requirement for the ORACLE.

3.2.4 Imperfectness of memory and memory decay

For humans, episodic memory is imperfect, as not every aspect of an event is remembered in the first place, and recall of that memory alters it. Furthermore, memories are forgotten over time if not recalled enough.

For the ORACLE, neither property is useful. If memories were to decay over time, the building of a corpus, as well as insight into the development of the child is lost. However, imperfectness of memories arises automatically by things such as errors in parsing, the design choices made for encoding of episodes and the way a child tells Charlie about events in their life. Except for these cases, however, no effort should be taken to incorporate memory decay or imperfectness.

3.2.5 Temporal indexation of memories

While the literature is unclear on how temporal indexation is achieved in the human brain (is there merely some mechanism to relate one event spatially to another, or is there some sort of exact internal clock), awareness of the temporal order of events is useful, if not necessary.

By indexing events temporally, the PAL system can reason not only using similar episodes, but on the order of episodes as well. This may become necessary when the system is able to detect a pattern, but this pattern changes over time (see design criterion 4). Furthermore, design criterion 5 requires the use of the affective state of the child, which is subject to change over time. By using a form of temporal indexation of episodes, the affective state of the child at the time the event occurred can be determined.

Temporal indexation is therefore a useful property of the memory system in the ORACLE.

3.3 Method

Now that the formal specifications are in place, the literature can be used to define an approach for designing the ORACLE.

First of all, since the resulting model should work with the PAL system and extend the previous research of Schreuder Goedheijt (2017), it will be implemented as an ontology in the HFC database.

Kearsley (1976) provides insight in when and why questions are asked. He

suggests people keep an internal knowledge base that is extended and corrected when new information is learned. When new information leads to a contradiction in the knowledge base, questions may be used to strengthen or weaken contradicting facts. Further more, questions can be used to acquire new information to grow the knowledge base. An approach using a knowledge base is taken by Han et al. (2015) and Bang et al. (2015) as well. In the case of Bang et al. this even serves as a way of bonding in the long term, since topics of interest are remembered over sessions. These knowledge bases seem to be focused especially on small talk, as the questions do not serve any purpose other than keeping the conversation going.

Schreuder Goedheijt (2017) initially based his representation of an episode on the 5W1H model suggested by Han et al. (2013), before they changed to triple extraction. They used their model only to ask questions right after a user utterance, so no notion of long-term interaction or memory was present in that system. However, the 5W1H corresponds neatly to the six frames of reference for the asking of open questions suggested by Kearsley. *Space* corresponds to the "where", *time* corresponds to the "when", *properties* correspond to the "what", *causes* correspond to the "why", *procedures* correspond to the "how" and *roles* correspond to the "who". Furthermore, Han et al. (2013) already include sentiment in their approach, which is a requirement for the current system as well. Not only does Han et al.'s approach translate to the previously determined requirements for encoding a single episode, it fits within the vision of Schreuder Goedheijt. Since the current research is a continuation of that of Schreuder Goedheijt (2017) and the 5W1H approach fits within the specified criteria, the 5W1H model will be the basis of the structure for a single episode.

Schreuder Goedheijt used an abstract representation of the 5W1H model, where each of the properties is represented by one or more predefined classes in the ontology. This requires each possible value for these properties to be predetermined and encoded in the ontology. This worked well for his approach, where the method for formulating a question using the dialog manager was equally limited, but no longer holds when potentially open-domain information in the form of natural language serves as the source for an episode.

Instead, the more direct approach also used by Han et al. (2013) in their original research will be used. With this approach, a sentence uttered by the user is split in the frames of reference and sentiment mining is applied to the original sentence. For example, the child might tell Charlie:

> "The teacher told me I could not have sugar at school yesterday."

This sentence can be split to 5W1H as follows:

Who	The teacher
How	told me
What	I could not have sugar
Where	at school
When	yesterday
Why	$\perp$
Sentiment	neg

From the instantiation of the model given in this example, it becomes clear right away that one value is missing before the episode would be completely annotated: the *cause*.

3.3.1 Question formulation with dialog management

In the most simples form, the robot could directly ask the most simple question "Why?" to determine the missing value. Although the formulation of the question itself is not the aim of this research, what will follow is a small discussion of how this may be attempted, to suggest that leaving dialog management outside the scope of this research does not entail this method is not achievable with the current state of the art.

As a direct response to the statement the context of simply asking "why" may be understood by the conversation partner, but when referring back to this sentence in a later episode, the context should be provided in the question. The context is provided by the frames of reference that do have a value. A *why*-question can be constructed by filling in these frames. For example:

"Why" + who + how + what + "?"

Filling in the frames of reference directly would provide the following question:

"Why The teacher told me I could not have sugar?"

This question clearly is not grammatical. In fact, this is simply the core of the wh-inform statement provided by the child and enclosed within questioning word and question mark. Further more, the perspective of the question is still that of the child. However, methods exist for rewriting an inform statement to question form. One example is that of Eliza (Weizenbaum, 1966), which matches common patterns in inform statements and uses rules associated with those patterns to rewrite the sentence to a question. Using such patterns, the question could be rewritten to:

"Why did the teacher tell you you could not have sugar?"

Now, this method of rewriting informative statements to interrogative form does not incorporate any understanding of what is being said by the dialog manager. This is not a problem, since the intelligence of the question lies in determining what to ask, not how to ask it.

With an increasingly advanced dialog manager, the intelligence behind formulating a question can be increased. It may even be possible to use the context without directly paraphrasing the child's original informative statement, if concepts can be sufficiently understood by a dialog manager. The current approach leaves open how the final question is formulated.

3.3.2 Ontology engineering

The ORACLE will be a model implemented as an ontology. As described in Section 2.2.1 on page 12, the PAL project makes use of seven separate ontologies, which are all connected through an eighth. In practice, the ontology that describes the PAL domain itself consists of various other ontologies working together, each describing a part of the domain. One of these ontologies is the one where the episodes of the Episodic Memory Manager are defined and stored. For this research, it will be that ontology which will be extended with new classes and relations to incorporate the ORACLE. To be able to use single queries over the complete set of ontologies within the PAL project, this ontology will be connected to the other relevant ontologies using equivalence relations on classes and properties in the `PAL` ontology. For this approach to work, some classes and properties that are already in other ontologies need to be added as phantoms, so the classes and properties from external ontologies have something to be equivalent *to*. This is in line with the design strategy maintained in the PAL project (Krieger et al., 2016).

3.3.3 Evaluation

After the ontology has been developed, it will be evaluated by determining whether it meets the design criteria specified at the start of this chapter. Design criteria 1, 2, and 6 are verified by the structure of the model itself. Design criterion 1 will be a consequence of the other criteria; if the model can be used to store, remember, and reason over episodes, this criterion has been achieved. An approach for criterion 2 has been given earlier in the method section. If this approach is successfully implemented in the ontology, this criterion will have been met as well. Design criterion can be verified is met if the child utterance can be stored in relation to the episode it results in without loss of information.

The other four criteria can be verified by showing the information specified in

those criteria can be extracted from the model using queries. To do this, mock-up data will be added to the HFC database by instantiating classes and properties and queries will be given that return the specified data. Since methods to parse child utterances and formulate questions are assumed to exist for this research, providing queries will suffice to show the potential of the model, provided the context of the episode, the topic of the question to ask an the type of question to ask can be easily determined by a future dialog manager using the query results.

The HFC query language is a proper subset of SPARQL. This means some functionality is not available in the former that is present in the latter. Most prevalent is the lack of an `OPTIONAL` operator in the HFC query language. Queries take the form of triples – or in the case of HFC, where assertion time is part of the tuple, *4-tuples* – concatenated together. Each item in the tuple is either a defined class, property or individual in the database, uniquely identified with an URI, or a variable, which starts with a question mark. The result of the query is the set of classes, properties or individuals that can unify with those variables. The `OPTIONAL` operator tells the query language to include the result, even if the variables enclosed within the `OPTIONAL` parameter cannot be unified with something in the database. Without this parameter, at least two queries are necessary for each variable otherwise designated as optional: One where triple with the variable is present, and one where it is not.

This is not a limitation of HFC in the context of the PAL project, as most queries are automatically constructed by the code depending on the current situation and the results of previous queries. The queries generated by the PAL system are often relatively compact and in some cases, using multiple queries may be more efficient computationally. However, to demonstrate the classes in an ontology form a single graph through their properties, rather than a number of distinct graphs, single queries will be given where possible.

The mock-up data which will be added to the HFC database will be based on the following examples. In these examples, the expected behavior of the ontology, as well as the required results of queries will be detailed.

3.3.3.0 Example 1

The child opens the MyPAL application. The robot's avatar greets the child and asks them what they did today. The child tells the robot: "I played sports today at school." This utterance is parsed and stored in the 5W1H model in the ontology:

Who	I
How	played
What	sports
When	today
Where	at school
Why	⊥
Sentiment	⊥

The system queries the ORACLE right away and learns two fields in the 5W1H episode do not have a value: Why and, assuming the sentiment cannot be detected by the sentiment mining algorithm, Sentiment. These two missing fields, along with the available context (the values of the other fields), should be available to the dialog manager by means of queries. These queries provide the functionality of the 5W1H model suggested by Han et al. (2013) and simultaneously show that design criterion 3 is met, as sentiment is encoded and used in this approach.

3.3.3.0 Example 2

Building on example 1, imagine the PAL actor asks a question to determine the sentiment of the child regarding the episode, and the child indicates they liked school. The next day, the process repeats: the child tells the avatar in the MyPAL application again they went to school, again Charlie asks the child how they liked school, and again the child indicates school was nice.

If the child tells the avatar the same thing yet again, the model should allow generalizing that the child likes school. Assuming all else to be equal, the question should not be repeated a third time to avoid repetitive behavior.

If a single query can be constructed that can detect similarity between the most recent episode and the previous episodes, and returns that missing sentiment value, design criterion 4a is met. This inferred information could subsequently be added to the ontology using a simple `INSERT` query. This assert will not be part of the model, however, to assure constraints on this reasoning mechanism (such as minimum number of source episodes, how exactly similarity is defined, etc.) can be specified in the code constructing the queries. This is something that does not change the model and may depend on different requirements of the system. For the current research it suffices to show a query exists that can achieve this generalization.

3.3.3.0 Example 3

Now assume a fourth day, where the child greets the avatar with different information: “Everything at school was stupid today.”

The sentiment mining and parsing algorithms fill in the 5W1H model as follows:

Who	$\perp$
How	$\perp$
What	everything
When	today
Where	at school
Why	$\perp$
Sentiment	neg

From the model it should then be possible to infer the similarity of this episode to the three episodes from examples 1 and 2 on the basis of the similarities of the When and Where tags. The dialog manager should then be aware that the frames of reference What and Sentiment are contradicting previous episodes and ask a question using one or both of these frames of reference. For example, "Why was it so bad at school today?" or "Did you not play sports today?"

If a single query can be shown to exist where the contradicting values of similar episodes are returned, design criterion 4b is met.

3.3.3.0 Example 4

For design criterion 5, an example is required that uses the affective state of a child. To evaluate design criterion 5, the mock-up originally used by Sternheim (2017) to verify the ESMAS will be used. A single query should show the mock-up emotional state from the ESMAS can be associated with a mock-up episode from the ORACLE. If such a query exists, it is shown the ESMAS ontology is successfully integrated with the PAL system and the ORACLE, and design criterion 5 is met.

Chapter 4
Implementation

The ontology was implemented using Stanford Protégé 3.5. Newer versions of this software are available, but due to more recent versions of the OWL language these versions use, they are not compatible with the PAL Project.

Since the current ontology is an extension of the Episodic Memory Manager, the classes required for the current ontology were added to the `EPMEM` ontology of Schreuder Goedheijt (2017). Schreuder Goedheijt's ontology revolves around a central `Episode` class, which knows three subclasses: `GoalEvent`, `TimelineEevent` and `DiaryEvent` (see Figure 2.9 on page 27). The former two refer to the types of episodes already implemented within the PAL system, namely episodes that refer to changes in goals or their progress (e.g. new goals added or quiz too difficult) and episodes that refer to information entered in the MyPAL timeline (e.g. went to school or played sports) respectively. The class `DiaryEvent` was exclusively reserved for future work, where episodes can refer to daily activities relayed by the child to PAL actor through open text, that is, not through an interface in the timeline which determines the context of an episode.

However, Schreuder Goedheijt original structure for an episode no longer fits within the newly envisioned 5W1H structure, so the first update to the existing ontology was to label the `DiaryEvent` class as 'depricated'.

The ORACLE will be built around a class called `SocialEpisode`. The core structure of this social episode, with the 5W1H properties and sentiment associated with it, was already discussed in the method section of the previous chapter. The rest of the ontology was extended from that core.

For the sake of clarity, the implemented ontology will be discussed in three parts. First, the formalization of the core episode structure will be discussed. After that, the structure for storing child utterances will be detailed. Finally, it will be explained how affect was integrated in the model.

Whenever it is said an equivalence relation between a class from this ontology and that of an external ontology was added, it is to be understood that relation

was added in the PAL ontology and not in the current ontology itself, in a similar way as shown in Figure 2.5 on page 14.

4.1 Episode core

The core of an episode will be the SocialEpisode class. This class corresponds to a single event, which may be described in natural language by a child. It was already discussed how a sentence can be broken up into the six 5W1H properties using the method of Han et al. (2013). However, not every event can necessarily be described in a single sentence. In fact a single utterance of the child may consist of multiple sentences or subordinate clauses.

Consider for example a child who tells the robot they have been to a birthday party and what happened at that party:

> "I went to a birthday party yesterday. I could have cake because I took insulin and after I measured, my values were perfect!"

This story concerns a single episode in the child's life, but should be split up in three sentences to ensure each of the 5W1H properties has only one value: "I went to a birthday party," "I could have cake because I took insulin," and "after I measured my values were perfect":

Who	I	I	I
What	birthday party	cake	perfect values
Where	⊥	⊥	⊥
When	yesterday	yesterday (inferred)	yesterday
Why	⊥	took insulin	⊥
How	went	could have	measured

4.1.1 Parsing uncertainty

A few decisions have been made for this instantiation of the 5W1H slots in this example. A small detour will be taken to explore these decisions.

First of all, "birthday party" could just as well refer to a place (Where), since it is something the child says they went *to*. It is instead interpreted as an activity (What), since a birthday party can take place *somewhere*. Interpreting "birthday party" as an activity, rather than a place is a design choice but does not make another interpretation wrong. The current design choice allows questions such as "how did you enjoy the birthday party," and "Where did the birthday party take

place." These types of ambiguity may arise more often but different interpretations usually only results in a different set of potential questions, rather than an invalidation of the model.

Another interpretation decision for this example was to leave the Why slot for the last column in the table empty. With an algorithm that is able to understand context and references beyond clause boundaries (e.g. as used in Hahn and Romacker, 1999) it could be understood that the Why, explaining why the measured values were perfect, should refer to "took insulin" from the preceding clause.

4.1.2 Episode parts

However, the point of this exercise was to show an episode may not fit within one unique 5W1H model. To account for this possibility, an episode in the ontology consists of one or more `SocialEpisodePart`'s. Each episode part should encode exactly one sentence or subordinate clause. An episode part is connected to the central episode it belongs to by the property `hasEpisodePart`.

Figure 4.1 shows the structure of an episode in this ontology[1]. Each of the 5W1H properties is connected to the `SocialEpisodePart` through a `has[WH]Tag`. None of the 5W1H tags is required to have a value, but they should not ever have more than one value.

Because the classes for all the 5W1H properties, including sentiment, have a lot in common, it is useful to define a superclass. The class `Property5W1H` has all the 5W1H classes as its children. Through inheritance, any relation that is defined on this superclass automatically also apply to the 5W1H classes. The relations in question are described in the next section.

4.1.3 References to other ontologies

In Figure 4.1, a few other classes are shown in a darker color. These classes are placeholders, which already exist in other classes within the PAL project.

First of all, any episode can be uniquely associated with the specific child who told it using the `sharedWith` property. Children are added as instances to the `Child` class, which is itself a subclass of the class `Actor`.

The `Actor` class in the `Domain` ontology contains more subclasses than just the one for the children. Other classes are those for the treating doctor, friends, and family members, such as parent or sibling. The `Domain` ontology also defines relations between these various actors. Friends, treating doctors, and family members

[1] A legend for this graph and others in this chapter can be found in Figure A.1 on page 80 in Appendix A

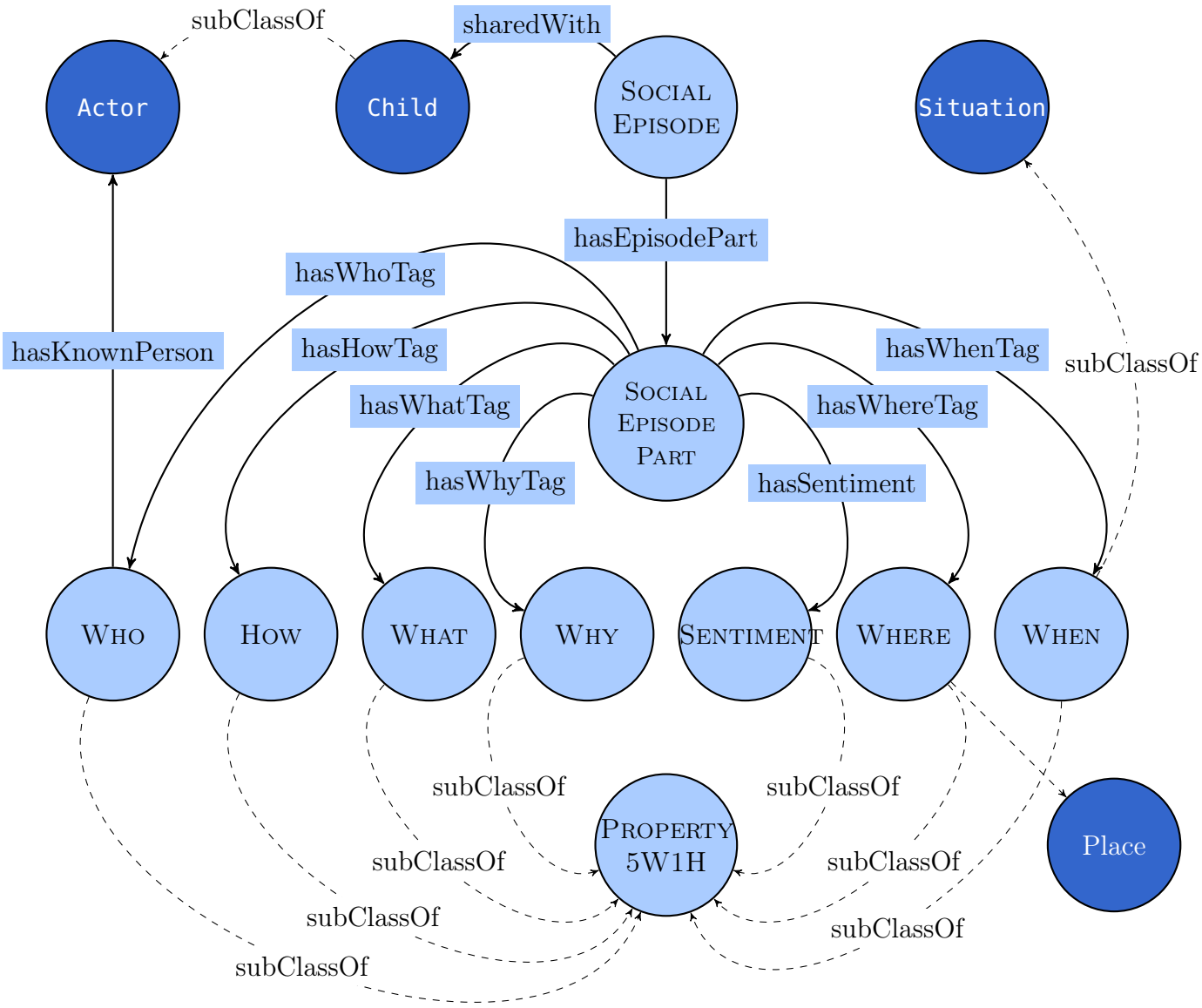


Figure 4.1: A representation of the core structure in the ontology: A SocialEpisode contains one or more SocialEpisodeParts, each of which has all the 5W1H properties as well as sentiment associated with it.

that are known to the system exist as instances in the HFC database and because of the `Domain` ontology it is known what the relation of those actors is to the child. Since it may be a real possibility these are also the people the child will often tell the robot about, the property `hasKnownPerson` can link an individual in the Domain ontology to the `Who` class of an episode part.

Lastly, the WH-classes `When` and `Where` can potentially be annotated with more

specific contextual information than the exact words from the sentence alone. For example, the word "yesterday" can automatically be translated to a time frame (somewhere between 12:00 am and 11:59pm on the date before that of today). Similarly, the `Where` can mark a specific location.

Since both concepts already know a class in the Upper ontology, the choice was made to reuse that previous work, rather than redefine them for the current ontology. For that purpose, the 5W1H property `Where` has been made a subclass of the Upper class `upp:Place` and the `When` a subclass of `upp:Situation`. This also allows the PAL system to reason over those two 5W1H properties using mechanisms already in place within the system.

4.2 Representing user input

One important design criterion was that the child's unaltered text could be stored in the ontology. This text should further be stored in relation to the episode that is extracted from that text. In this section, the part of the ontology which handles storing this data is discussed.

Figure 4.2 shows the visualization for this part of the ontology. The class `Property5W1H` already featured in Figure 4.1 and has therefore been marked a lighter color.

Every time a child tells the robot something, this should be stored in the database. In this ontology, the class `Utterance` was created for that purpose. An utterance has a string value associated with it, in which raw text can be stored. The actor who uttered the utterance is associated with the class `isUtteredBy`.

However, in the previous section it was shown that an utterance can contain multiple sentences or subordinate clauses. In this ontology, both the complete user utterance and the sentences and subordinate clauses that utterance splits up in are stored. The class `Utterance` therefore has two subclasses, both inheriting its properties: `FullUserUtterance` and `FullUserSentence`. The user's utterances are stored in these subclasses, rather than in the `Utterance` class itself.

The first class stores the complete unaltered user utterance. In the case of the example from the previous section, this will be: "I went to a birthday party yesterday. I could have cake because I took insulin and after I measured, my values were perfect!"

This utterance is split up in its three sentences[2]:

1. "I went to a birthday party yesterday"

[2]or subordinate clauses, if a sentence consists of multiple subordinate clauses. In the remaining of this chapter, the term sentence will be used to refer to either.

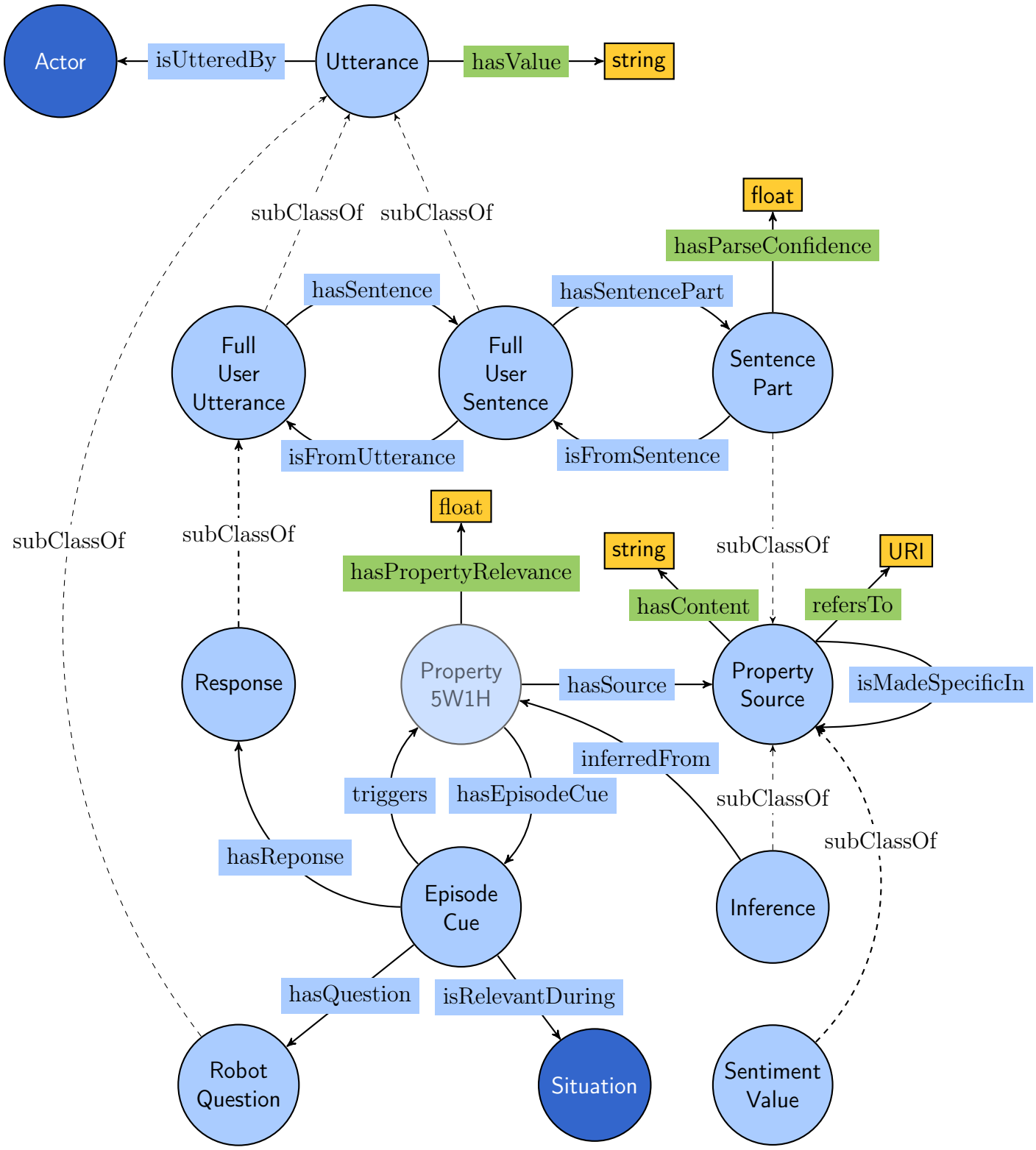


Figure 4.2: The user utterance is first split into complete sentences and then further into sentence parts. These sentence part can be the source of a 5W1H property, but an inference, containing references to one or more 5W1H properties of previous episodes may also be the source of a 5W1H property.

2. "I could have cake because I took insulin"

3. "After I measured, my values were perfect!"

Each of these sentences is in turn stored on the `FullUserSentence` class.

The separate sentences are connected to the utterance they originate from through the `isFromUtterance` relation. The reverse of this relation is called `hasSentence`, and allows all sentences belonging to a single utterance to be found.

4.2.1 Property source

In Figure 4.1 the seven 5W1H properties of an episode part were visualized. These properties encode what information a specific part of the sentence contains.

Each of the 5W1H properties can one of three subclasses of a `PropertySource`. This source contains a reference to the exact value of the 5W1H property. These values provide context about the episode to the dialog manager, which can be used to formulate a question. If a 5W1H property does not have a value, the dialog manager can ask information about that property.

The string value of the 5W1H property is associated to the property source with `hasContent`.

4.2.1.0 Types of property sources

The class `PropertySource` itself is never used directly. Instead, the real source for a 5W1H property is one of its subclasses: `SentencePart`, `Inference` or `SentimentValue`.

The `SentencePart` class is instantiated when the value of a 5W1H property is taken directly from a user utterance. The *content* of a sentence part can be words or short groups of words such as "I", "birthday party" or "yesterday," each of which containing the information about on of the seven 5W1H properties. The sentence part is connected to the sentence it was extracted from through the `isFromSentence` property. This completes the chain linking a full user utterance to the episode it describes.

A sentence part can have a parsing certainty associated with it. In Section 4.1.1 on page 41 an analysis was given of how some words in a sentence might give information on multiple 5W1H properties. If a parsing algorithm supports it, the probability the correct 5W1H property was chosen by this algorithm can be stored. This information could be used by a dialog manager to give 5W1H properties with a higher parsing certainty priority when using them for question formulation.

The `Inference` class is used when a 5W1H property is inferred from preceding episodes or episode parts on the basis of similarity. These inferences are made

on the assumption that if a pattern exists, that pattern will likely apply to the current situation if nothing directly contradicts that assumption. Inferences in this ontology are therefor inductive and not deductive.

The `Inference` class has an object property called `inferredFrom`. This property contains a link to one or more 5W1H properties of *another* episode part: the one where this value was inferred from.

By defining a separate class for properties that were inferred, these properties can easily be distinguished from 5W1H properties that were extracted directly from user input. In the case a conflict arises, because a child tells the robot something that directly contradicts an inferred property, the child's utterance should be leading, since inferences may be based on flawed assumptions. Alternatively, a question could be asked to determine which value is the correct value for that 5W1H property.

Lastly, the `SentimentValue` class serves as the source of the 5W1H property `Sentiment`. The sentiment itself ('`+`', '`-`' or '`0`') is associated with this class through the `hasContent` data property. This value of a sentiment is further discussed in Section 4.3 on page 50.

4.2.1.0 Recognized entities

The `PropertySource` class contains a property that allows linking all its three subclasses to an external URI. This property is called `refersTo`, is optional, and depends on the capability of the parser.

If the parser allows it, this property is used to store a reference to an instance or class from another ontology or knowledge base about the interpretation of the words in the `hasContent` property. The URI could be from the domain ontology this model is used in, in which case it refers to a well-known class or entity within that domain. This allows all domain knowledge about this URI to be applied to the 5W1H property value as well.

The URI could, however, also point to a class, relation or instance of an external knowledge base, such as DBPedia[3] or WikiData[4]. By storing such references, the full power of knowledge bases could be exploited to formulate more informed questions to the children.

For example, the PAL domain knows various sports, such as football, tennis and waterpolo. If a child tells the PAL actor they like to play dodgeball, which is not in the domain knowledge of the PAL system, but which could be recognized as an entity in an external knowledge base[5], the information from the external

[3] `https://wiki.dbpedia.org/`
[4] `https://www.wikidata.org/`, previously FreeBase
[5] `https://www.wikidata.org/wiki/Q473719`

knowledge base could be used to determine what the child is talking about. In the case of dodgeball, WikiData tells us it is an `instanceOf` the class "sport", which *is* known in the domain. This information can be used to ask an informed question that works for any other sport in the domain and thus allows the system to respond intelligently to a topic of interest of the child while that topic is not in the system's domain knowledge.

Storing the URI to an external knowledge base further allows question formulation on the basis of interesting properties of the recognized entity or about well known instances of the same type for that entity, as was the original approach of Han et al. (2015) and Bang et al. (2015). Do note that this property may have multiple values, as a class or instance may be present in multiple knowledge bases.

4.2.1.0 Specifying what is meant

Dodgeball is a specific sport, so knowledge of what dodgeball is was used in the previous example. However, the child could just as well have said they played sports, without mentioning the specific sport. Because the PAL Domain ontology contains multiple instances for sports, the system can know the child likely played a *specific* sport. A question could be formulated to determine what the exact sport was a child played.

In any situation where the PAL actor asks the child to further specify a concept, this concept is stored in a new property source and connected to the more general source using `isMadeSpecificIn`. This property is made *transitive*, meaning that if property source A (e.g. sport) is made specific in B (e.g. teamsport) and B is made specific in C (e.g. dodgeball), then C is also a more specific instance of A. In fact, C is the most specific property source of A that is currently known. Marking the `isMadeSpecificIn` property as transitive allows recursively finding all nested specifications in a single query.

4.2.2 Storing the answer to a question

When the robot formulates and asks a question concerning an episode, or, through inference, a set of episodes, the child is expected to answer that question. Usually, that answer will be in the form of a wh-inform statement, which provides new information on an episode, or is the basis for a new episode.

Since child's answer is also an utterance, the answer should be stored as well. Of course, such an answer can be treated as any other utterance, but it would make sense to add a way to trace a child's answer back to the question it answered and the episode that was used to formulate that question.

Each 5W1H property is given a `EpisodeCue`. This cue both determines when

a question on a specific property can be asked, and connects the child's response to the PAL actor's original question and the 5W1H property itself.

4.2.2.0 When to ask a question

The class `EpisodeCue` defines when it is relevant to ask the child for the value of the 5W1H frame of reference it is connected to. Such a question can be asked directly after the user utterance has parsed and it becomes clear a specific 5W1H property is missing. However, there are situations when such a question should not be asked right away.

The example utterances of the child so far have all used a past tense. However, the child could also inform the PAL actor of an upcoming event, for example that the child has been invited to stay over at family. If the PAL actor would, for example, ask the child if they enjoyed the sleepover, that question should be delayed until after the event.

Being able to specify a time frame frame during which it is relevant to ask a question about a 5W1H property would be useful. This functionality is exactly provided by the Upper ontology, through the `Situation` class. This class describes a time frame with, amongst others, a start and end time. This could tell the dialog manager not to use this episode cue for a question just yet, if the start time is in the future, or not any more, if the end time is in the past.

Of course, either or both the start or end time can be left unspecified, in which case they should not be used for the relevance check.

4.2.2.0 Storing both question and answer

The episode cue has two other classes associated with it. The first class it the `RobotQuestion`, which allows the system to store the question the dialog manager formulated and which was then asked to the child. This class is a subclass of `Utterance`, and the utterer is the PAL actor.

The second class, `Response`, is a subclass of the `FullUserUtterance`, to distinguish it from a user initiated utterance, and contains the answer of the child. Through inheritance, it contains the same properties as the `FullUserUtterane` class, meaning it can be split into sentences and further split into sentence parts, which in turn serve as the source for 5W1H properties.

If the child's answer gives information about the single 5W1H property the question was asked about, the property source will be associated with the same 5W1H property the epside cue belongs to. If the child's answer provides completely new information on the episode, a new episode part will be instantiated, which in turn can be the reason for asking yet another question to the child.

4.3 Adding affect

The affective state of the child is expressed in the ontology of affect, the ESMAS (Sternheim, 2017). This is a separate ontology and should be kept as such. Integrated within the PAL system and, by extension, the current ontology for episodic memory is done by defining equivalence relations in the PAL ontology.

The ESMAS does not take sentiment values – which are important to the PAL system – into account and as was seen in Section 2.3.2 on page 17, sentiment is not directly expressible as either emotion or mood. This means sentiment should be added to the current ontology. The concept of sentiment had already been visualized in the previous section, but has not yet been fully described.

4.3.1 Sentiment and sentiment value

For the current ontology, it was explicitly decided to make sentiment a subclass of `Property5W1H` (see Figure 4.1 on page 43). This has the advantage it can be treated as any other 5W1H property when querying the database. However, the sentiment value does not correspond directly to a part of the sentence. If the child tells the robot they are happy, the word "happy" is the indicator for the sentiment, but the sentiment value is '`+`'. Furthermore, sentiment is not necessarily a uniquely identifiable part of the child's sentence itself, but may also be an abstract value concerning the sentence as a whole.

For this reason, the class `SentimentValue` was made a subclass of `PropertySource`. This ensures that, while the class `Sentiment` is treated as a 5W1H property, `SentimentValue` can be used both as the *source* of that sentiment, as well as be associated with a user utterance that is not directly related to an episode.

Currently, sentiment analysis provides only a signed valence ('`+`' for positive sentiment, '`-`' for a negative sentiment, and '`0`' for neutral or undetermined sentiment), but research is currently ongoing in analysis methods for finding the *topic* of the sentiment. For example, when a child tells the robot "School was fun today," not only can the system determine the sentiment of this sentence is positive, but also that this positive attitude is directed at the topic *school*. Whenever this topic can be determined, it should have a place in the ontology, directly related to the sentiment it belongs to.

Figure 4.3 visualizes the `SentimentValue` class. The class inherits the possibility to add a string value through the `hasContent` property from its superclass (see Figure 4.2 on page 45). This string value can be '`+`', '`-`' or '`0`'.

The `SentimentValue` class can also have a topic, which is associated with it by the property `hasTopic`. This topic is itself stored in a subclass of `PropertySource`.

If the topic refers to a word in the sentence, the topic will be a `SentencePart`. The topic could also be inferred from other episode parts on the episode, in which case it will be instantiated as a `Inference`.

By storing the sentiment topic as a property source, it not only can take a string value, but be related directly to the sentence that topic comes from, and have an URI describing that concept related to it.

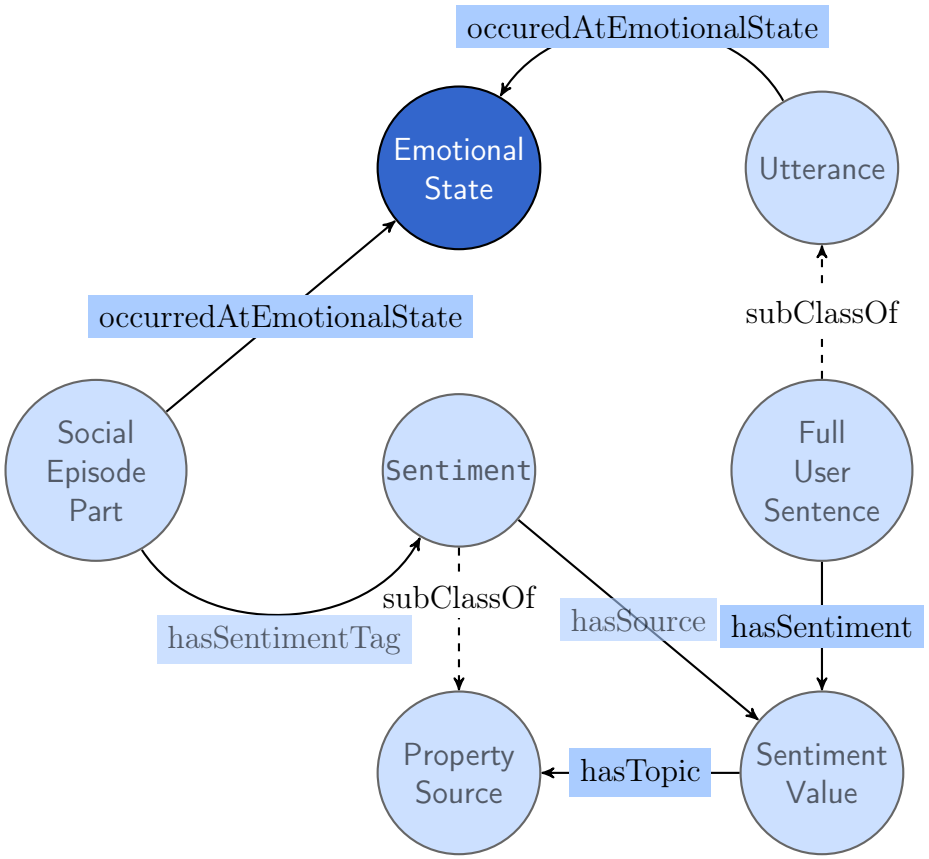


Figure 4.3: The sentiment and emotional state comprise the affective state of the child. The logic for the model of affect is located in the ESMAS (Sternheim, 2017), a separate ontology that is connected to the current by defining equivalence relations.

4.3.1.0 Using sentiment

There are two classes in the ontology which can have a sentiment value associated with it. These are the `SocialEpisodePart` and `FullUserSentence` classes.

As was discussed in Section 2.3.2 on page 17, sentiment should be treated as an attitude towards something and one utterance may contain multiple sentences, each with their own sentiment. Sentiment analysis should therefore only be performed on the level of individual sentences, after a complete user utterance has been split. For this reason, individual `FullUserSentence` instances can have a sen-

timent value associated with them. Sentiments are associated with a user sentence through the property `hasSentiment`.

Episodes can have sentiment associated with them through the 5W1H property reserved for this. This property has a `SentimentValue` associated with it as a property source.

4.3.1.0 Integrating the ESMAS

The ESMAS can be connected to the other ontologies used in the PAL project, and to the current one for episodic memory, by means of a few equivalence relations, which are defined in the PAL ontology. In the rest of this chapter, it will be explained how these equivalence relations are defined. A visualization of all the relevant classes in the various ontologies used by the PAL project is given in Figure 4.4.

The ESMAS contains a few important classes around which the rest of the ontology revolves: `Mood`, `Emotion` and `EmotionalState`. The `Mood` and `Emotion` are both part of the `EmotionalState` through the properties `hasMood` and `hasEmotion` respectively. This means connecting the emotional state to the other ontologies is sufficient for being able to query mood and emotion as well.

The PAL `Domain` ontology already contains a class for mood, including instances corresponding to various moods that were envisioned a child could have. However, this class was just a placeholder until a better way to represent the emotional state of the child had been developed and is not used anywhere within the PAL project. This is fortunate, as this means the class `Mood` from the `Domain` ontology can be disregarded without loss of functionality, which is by far the easiest solution for replacing it. In Figure 4.4, this has been shown by marking the class as gray.

The affect ontology was added to the cluster of ontologies in the PAL project, by defining an equivalence relation between the `Agent` class in the ESMAS and the `Actor` class in the Domain ontology. This means that the emotional state of every actor, including all children and the PAL actor, but also family members and friends of the children, could potentially be modeled and are automatically associated with that agent.

The ESMAS ontology defines a property `hasEmotionalState` with the actor as the domain and the class `EmotionalState` as its range. By defining the equivalence relation between the actors of both ontologies, this property is automatically available to the actor classes defined in other ontologies. This way, the emotional state becomes available to all ontologies that make use of the `Domain Actor` class.

In the current ontology for episodic memory, where the ORACLE is implemented, `EmotionalState` was also added as a placeholder class. The reason for

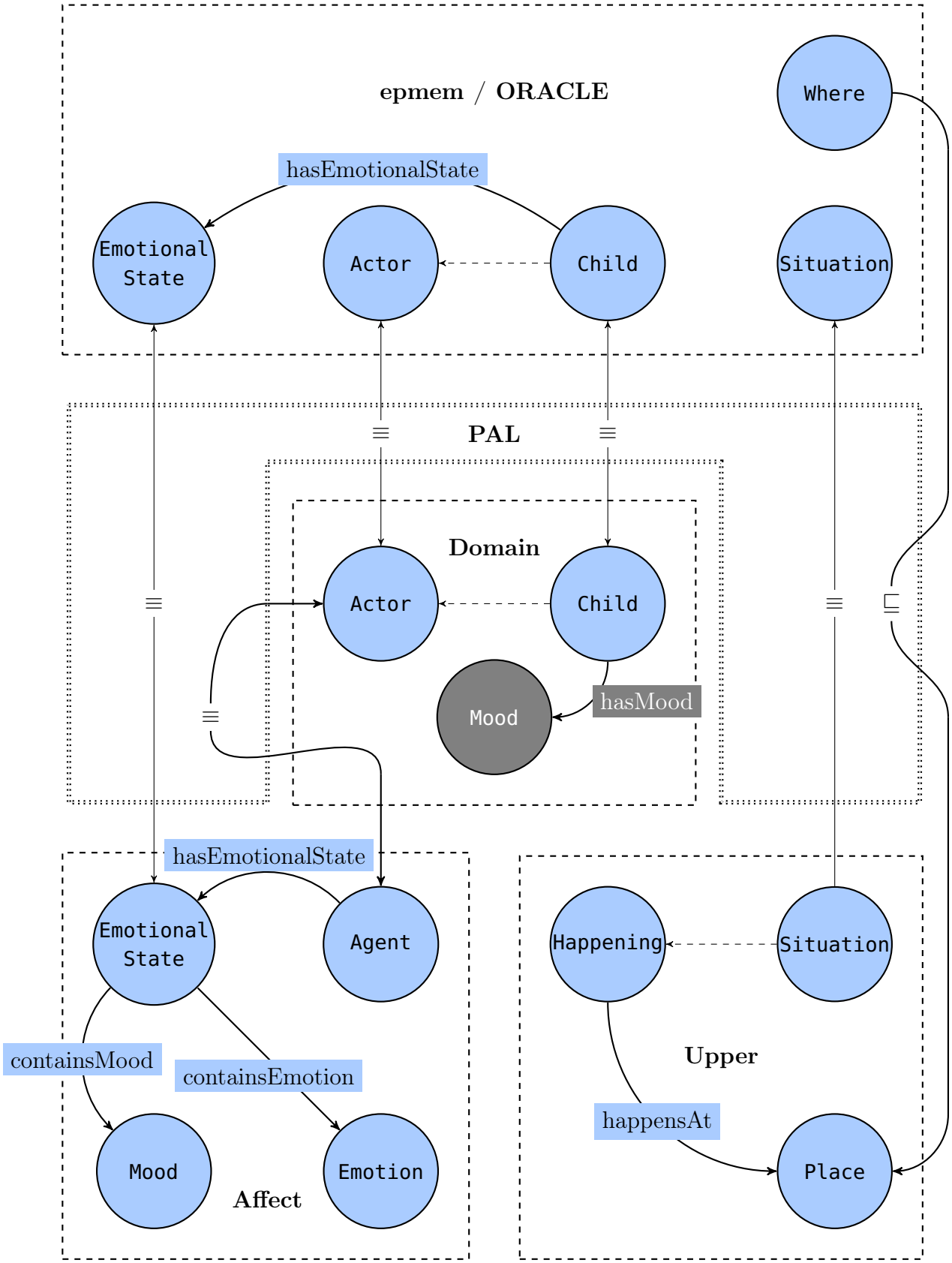


Figure 4.4: Various classes from the ORACLE ontology are connected to those of the ESMAS. These relations are all defined in the PAL ontology. The `Mood` class in the domain ontology is no longer required.

this is that the emotional state of an actor is subject to change over time. By associating it with any utterance, as well as with an episode part, the emotional state of the child at the time of an utterance or episode can always be determined.

The emotional state of the child could also be determined by using the assertion time in the HFC database, by finding the last emotional state that was asserted at the time the episode part of utterance was added to the database. This means that in essence, this addition is redundant. However, the added relations not only allow the emotional state for a specific utterance or episode part to be extracted, but also describe how these three classes should interoperate.

To unlock the full potential of the ESMAS ontology, a last equivalence relation between the `EmotionalState` classes of the ESMAS and the current episodic memory ontology was defined.

Chapter 5
Evaluation

In this chapter, the six design criteria specified in Chapter 3 will be evaluated using the method described in Section 3.3 on page 33. Next, the model will be compared to the features envisioned at the start of that chapter.

Some of the design criteria are validated by using queries. Of these queries, only the results are shown. The technical reader interested in these queries can find them in Appendix B. The queries in that appendix are grouped by the design criterion they are used to validate.

5.1 Design criterion 1

Design criterion 1 stated the model should exhibit some form of memory, and the type of memory exhibited should be analogous to the human *episodic memory*.

The ORACLE centers around the concept of an *episode*, inspired by human episodic memory. Episodes in the ORACLE encode stories of events that are told to the system through natural language by the user and thus relate to events from the users life. Memory is exhibited by the ORACLE by its capability to both store and retrieve these episodes.

In Section 3.2 on page 31 episodic memory was described by means of a number of properties, that distinguished episodic memory from other memory systems. The properties that were then said to be relevant for the ORACLE will shortly be discussed, to verify if the type of memory that is exhibited is indeed episodic.

5.1.1 Autobiographic memory

The matter of perspective of the memory was discussed in Section 3.2 on page 31. The perspective of the memory of the event is that of the child. This is successfully modeled using the `sharedWith` tag, which uniquely links an episode to the child who told of it. The complete event is partially related to the system by the child

and this is the event that is 'remebered' by the PAL actor and which the PAL actor reasons over. This satisfies the requirements regarding autobiographic perspective that were specified in Section 3.2 on page 31.

5.1.2 Subjective time

The matter of subjective time was translated to the requirement that episodes should be readily available whenever the system needs them and that it should be possible to place them in the context of the time they occurred. Retrieval of episodes is enabled my using an ontology database, where episodes are treated as sets of data. By means of queries, any episode can be requested from that database. An episode can be placed in the context of the time it occurs, since a child's emotional state at that time can be requested, and since the PAL actor's questions are linked to both the episode they were based on, and to the child's answers.

5.1.3 Imperfectness of memory and memory decay

Another requirement was that memories should *not* be imperfect and that memory decay should *not* be incorporated. Memory decay in software implementations, contrary to human memory, is something that needs to be added as functionality explicitly. Barring errors on a hardware level on machines that store the ontology database, memories will not decay using the current ontology, since this would mean explicitly deleting episodes. Imperfectness is only introduced to the memories in the ORACLE at instantiation, and originates in either an imperfect retelling of the event by the child, or by errors on parsing. Once those memories are instantiated, however, they no longer change. This is different from human episodic memory, where the very act of recalling alters the memory itself.

5.1.4 Temporal indexation of memories

A last requirement relating to the concept of episodic memory was that episodes are index temporally. In the ORACLE, episodes are embedded in time by means of the *assertion time* in the HFC database, as well as through `when` property of an episode episode part, which contains the time the event occurred. The assertion time allows a complete ordering of episodes on the basis of when they were added, since this time is precise to the level of microseconds. The ontology was set-up to consist of multiple episode parts, however. These episode parts also allow a complete ordering by means of assertion time, but this will be a different ordering than that of the main episodes themselves. This is because extra episode parts

could be added a significant time after the episode was instantiated in the model. Using the parsed value of the `When` property of an episode part, a partial ordering can be constructed on when the events the child talks about took place. These times are not necessarily specific, however ("yesterday" indicates a range, rather than a specific point in time), so no complete ordering could be constructed this way. Since a complete ordering for episodes and a partial ordering for the events these episodes describe exists, the temporal indexation of memories is achieved.

5.2 Design criterion 2

The second design criterion stated that the ORACLE should be able to encode informative statements by splitting them into 7 frames of reference: *who*, *what*, *where*, *when*, *why*, *how* and *sentiment*. In section Section 3.3 on page 33 an approach was given for modeling informative statements. A full description of how this model was incorporated in the ORACLE was given in Section 4.1. In Section 4.2.2 it was further described how the answer a child gives to a question of the PAL actor can itself be translated to an episode. So, the ORACLE allows encoding of informative statements, both elicited by the child on their own accord, and those that are answers to the PAL actor's questions.

5.3 Design criterion 3

The third decision criterion specified the ORACLE should be able to encode sentiment in such a way it provides extra information about the episode. The design decision was made to store sentiment as a 5W1H property related to an episode part. This means questions can be asked about sentiment, and that sentiment is a part of the episode context for other questions.

Sentiment values can also be assigned to a sentence part, relating it to the child's utterances. This allows sentiment to be stored even if no episode can be generated from a user utterance.

Queries where sentiment is used will be given for the other design criteria, but for now this analysis suffices to conclude this design criterion is met.

5.4 Design criterion 4

The fourth design criterion is perhaps the most important for the intended functionality of the ORACLE. The model has been developed around this criterion

first and foremost, and simply pointing to the model structure does not suffice to show the requirements given for this design criterion have been met.

Instead, as indicated in the method section, what follows is a number of queries to show the model actually allows the behavior specified in this criterion. However, it will first be explained how the examples given in the method section were added to the database.

5.4.1 Adding an episode

When creating instances in an ontology, similar to when classes and properties are generated, a name is given to that instance that, together with the URI of the ontology itself, uniquely denotes that individual. Names can be chosen to enhance interpretability if individuals are added manually, but in practice these names will be automatically derived from the class or property they instantiate.

The social episode that was manually added to the ontology is called `SocialEpisode_1`. This individual is uniquely identified with the URI `http://www.dfki.de/lt/onto/pal/episodicmemory.owl#SocialEpisode_1` or, since the prefix for the Episodic Memory ontology was set to `epmem`, as `epmem:SocialEpisode_1`.

A `SocialEpisodePart_1a` was added to the episode to encode the first example given in Section 3.3.3 on page 37. The utterance of that example was added to the database, and split into sentence parts accordingly, which were then associated with the corresponding episode tags of `epmem:SocialEpisodePart_1a`. Figure 5.1 gives a representation of how the first example looks after it has been instantiated in a database.

5.4.2 SPARQL query results

The most general interpretation of design criterion 4 is that, given an episode, the model should be able to provide a 5W1H property to the dialog manager to ask a question about and the context available on that episode. 5W1H properties that do have a source provide context, while those that do not have a source provide an opening for a question.

Table 5.1 shows the results of such a query. The SPARQL query, which can be found in Listing B.1, requests all episode parts for `SocialEpisode_1` and, of those episode parts, all property sources, regardless of whether they have a value or not. The episode cue is also requested, so the dialog manager can start constructing a question right away.

The prefixes have been left out of the values given in Table 5.1.

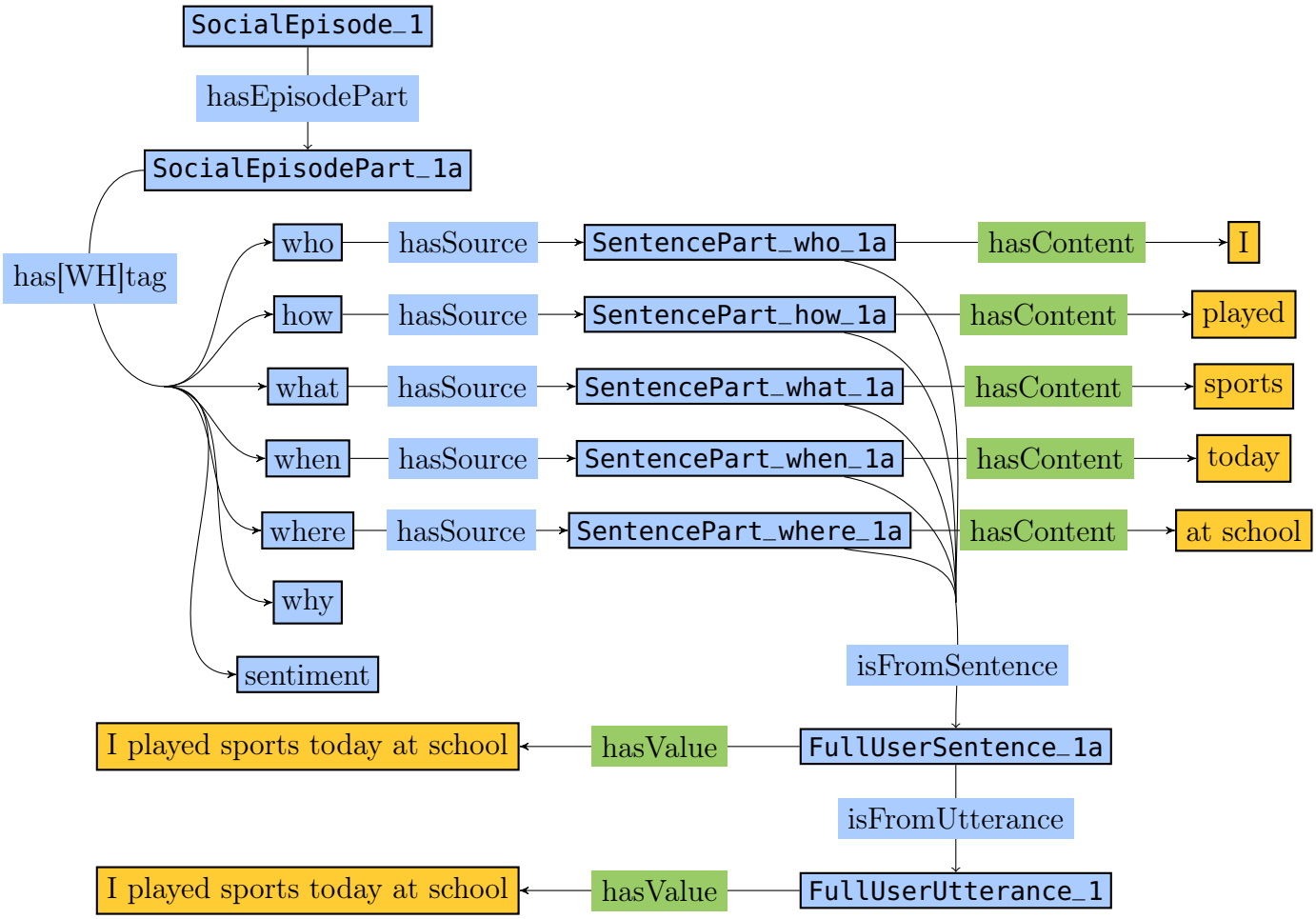


Figure 5.1: A mock-up social episode was added to the database. Visualized are the relations between the instances of classes, rather than the classes themselves.

Table 5.1: The result of a SPARQL query to find all 5W1H properties and their sources of an episode.

episodepart	tag	source	cue
SocialEpisodePart_1a	What_1a	SentencePart_1a_What	EpisodeCue_1a_What
SocialEpisodePart_1a	How_1a	SentencePart_1a_How	EpisodeCue_1a_How
SocialEpisodePart_1a	Sentiment_1a		EpisodeCue_1a_Sentiment
SocialEpisodePart_1a	Who_1a	SentencePart_1a_Who	EpisodeCue_1a_Who
SocialEpisodePart_1a	Where_1a	SentencePart_1a_Where	EpisodeCue_1a_Where
SocialEpisodePart_1a	When_1a	SentencePart_1a_When	EpisodeCue_1a_When
SocialEpisodePart_1a	Why_1a		EpisodeCue_1a_Why

5.4.3 First HFC query results

The same result can be achieved in HFC. However, since the `OPTIONAL` operator was used in the SPARQL query, and this operator is not present in the HFC query

language, two separate queries are required.

In the first query for the HFC database, again all episode parts of the episode are requested. Then, of all those episode parts, all 5W1H properties that have a source associated with them are requested. This filters out all 5W1H properties that could not yet have been instantiated, because the property source cannot be unified with that query.

The result of this query is shown in Table 5.2. The query itself can be found in Listing B.2. In the result table, the `epmem` prefix is left out wherever an individual name starts with a colon.

Table 5.2: The result of an HFC query finding all 5W1H properties of `SocialEpisode_1`, along with their values.

?episodePart	?t1	?tag	?t2	?source	?content
<:SocialEpisodePart_1a>	"0"	<:What_1a>	"0"	<:SentencePart_1a_What>	"sports"@en
<:SocialEpisodePart_1a>	"0"	<:How_1a>	"0"	<:SentencePart_1a_How>	"played"@en
<:SocialEpisodePart_1a>	"0"	<:Who_1a>	"0"	<:SentencePart_1a_Who>	"I"@en
<:SocialEpisodePart_1a>	"0"	<:Where_1a>	"0"	<:SentencePart_1a_Where>	"at school"@en
<:SocialEpisodePart_1a>	"0"	<:When_1a>	"0"	<:SentencePart_1a_When>	"today"@en

The assertion times, `?t1` and `?t2`, have the UNIX time stamp 0 for all rows in this result. This is because all instances were added before the HFC database was loaded, which meant that the transaction time is set to the earliest time that can be described in a UNIX time stamp: 0. If the triples were set by the PAL system, while HFC had already been loaded, the HFC database would automatically add a UNIX time stamp with the time of that transaction. For the current validation purposes, the exact value is not important, but it is good to realize that in real situations, transaction times would only ever be the same if either the assertion is a necessary truth (in which case the transaction time is 0) or if all triples were added simultaneously, using a single assert.

5.4.4 Second HFC query results

The second HFC query is meant to find all 5W1H properties on the episode that do not have a property source. For this query, the results from the previous HFC query are used. The query finds all 5W1H properties for a specific episode part, and then filters out all those properties which do have a property source associated with them. The results of this query are given in Table 5.3. The query itself can be found in Listing B.3.

Table 5.3: The the result of the HFC finding the 5W1H properties of epmem:SocialEpisodePart_1a that do not yet have a source associated with them.

?tag	?cue
<epmem:Why_1a>	<epmem:EpisodeCue_1a_Why>
<epmem:Sentiment_1a>	<epmem:EpisodeCue_1a_Sentiment>

Table 5.2 and Table 5.3 together contain the same values as the results in Table 5.1, but also show the assertion time.

5.4.5 Design criterion 4a

To verify if the ontology is able to infer values from similar episodes in its memory, example 2 given in Section 3.3.3 on page 38 was instantiated in the ontology, by copying epmem:SocialEpisode_1 three times. All instances relating to that episode were copied as well, and associated with the new episode. The integers indicating the social episode were incremented accordingly in the individual's names.

epmem:SocialEpisode_2 and SocialEpisode_3 were subsequently given a second episode part, which encodes the answer to the question "How did you like playing sports at school." A positive sentiment was added as the source for the 5W1H property relating to sentiment to both these new episode parts, and was also used to fill in the sentiment of the original episode parts, to mimic the updates to the ontology that should occur when the PAL actor and child are engaged in a short conversation during real use. The second social episode is visualized in Figure 5.2.

This mock-up data corresponds to the example given in Section 3.3.3 on page 38. The consequence of this update to the ontology is that epmem:SocialEpisode_2 and SocialEpisode_3 now have five tags that have the same value as epmem:SocialEpisode_4 (Who, How, What, When, Where) and one value for a 5W1H property that does not have a value on SocialEpisode_4.

5.4.5.0 SPARQL query results

A query was constructed in SPARQL to find these similar episodes, and infer the one missing 5W1H property value from those similar episodes. The query assumes an episode should have the same value on at least three 5W1H properties to be counted as similar, but this is an arbitrary number that can be adapted to what is required by the dialog manager. The result for the query is given in Table 5.4. In the results, it can be seen which value can be inferred. The 5W1H property that value belongs to, as well as the episode part is was inferred from, are also shown. The last column states on how many properties the given episode part was similar

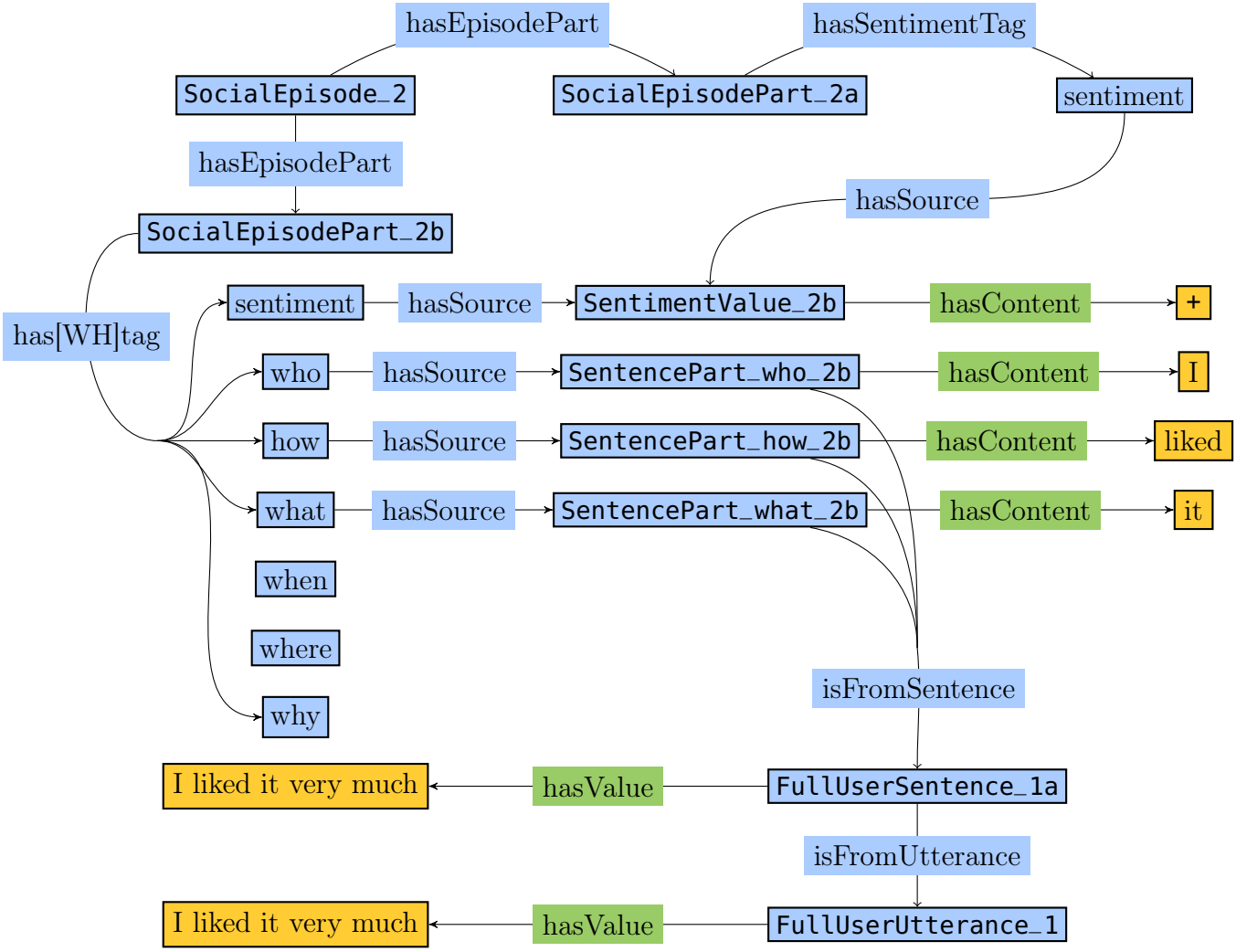


Figure 5.2: A second episode part is added, encoding the child's answer to the question "How did you like playing sports at school?" The `SentimentValue_2b` source is also added to the previous sentiment part on `SocialEpisode_2`, since the question served to find out the sentiment for that episode part.

to the episode part where this value was missing. The query itself can be found in Listing B.4

Table 5.4: The result of the SPARQL query finding all episode parts that have at least 3 similar properties as those of `SocialEpisode_4`.

similarEpisode	whtag	property5W1H	value	nSimilarTags
SocialEpisodePart_2a	hasSentimentTag	Sentiment_2a	+	5
SocialEpisodePart_3a	hasSentimentTag	Sentiment_3a	+	5

The last column shows how many tags are indeed the same. The `value` column shows the value from the similar episode part. This is the value that could potentially be inferred.

5.4.5.0 HFC query results

Again, to achieve the same result using the HFC query language, two separate queries are required. The first query finds all episode parts that are similar to that of `epmem:SocialEpisode_4`, along with the values these episode parts have in common. The result of this query is given in Table 5.5. The query itself is provided in Listing B.5.

Table 5.5: All episode parts with values similar to those of episode `epmem:SocialEpisode_4` are found with an HFC query.

?similarEpisodePart	?whtag	?property5W1H	?value
<:SocialEpisodePart_1a>	<:hasHowTag>	<:How_1a>	"played"@en
<:SocialEpisodePart_1a>	<:hasWhereTag>	<:Where_1a>	"at school"@en
<:SocialEpisodePart_1a>	<:hasWhenTag>	<:When_1a>	"today"@en
<:SocialEpisodePart_1a>	<:hasWhatTag>	<:What_1a>	"sports"@en
<:SocialEpisodePart_1a>	<:hasWhoTag>	<:Who_1a>	"I"@en
<:SocialEpisodePart_2a>	<:hasWhatTag>	<:What_2a>	"sports"@en
<:SocialEpisodePart_2a>	<:hasWhenTag>	<:When_2a>	"today"@en
<:SocialEpisodePart_2a>	<:hasHowTag>	<:How_2a>	"played"@en
<:SocialEpisodePart_2a>	<:hasWhereTag>	<:Where_2a>	"at school"@en
<:SocialEpisodePart_2a>	<:hasWhoTag>	<:Who_2a>	"I"@en
<:SocialEpisodePart_2b>	<:hasWhoTag>	<:Who_2b>	"I"@en
<:SocialEpisodePart_3a>	<:hasWhatTag>	<:What_3a>	"sports"@en
<:SocialEpisodePart_3a>	<:hasHowTag>	<:How_3a>	"played"@en
<:SocialEpisodePart_3a>	<:hasWhenTag>	<:When_3a>	"today"@en
<:SocialEpisodePart_3a>	<:hasWhereTag>	<:Where_3a>	"at school"@en
<:SocialEpisodePart_3a>	<:hasWhoTag>	<:Who_3a>	"I"@en
<:SocialEpisodePart_3b>	<:hasWhoTag>	<:Who_3b>	"I"@en

The prefix epmem has been removed in every value that starts with a colon.

The results from Table 5.5 are then processed by the code behind. This code finds all episode parts that have at least the minimal amount of similar 5W1H values and generates a new query for find all 5W1H property values on each of those episode parts. The results of this query can be found in Table 5.6. An example query, which finds all 5W1H property values for `epmem:SocialEpisodePart_2a`, can be found in Listing B.6.

By comparing the resulting values with those already present on `epmem:SocialEpisode_4`, the values that are missing can be inferred by the code.

Table 5.6: The result of Listing B.6 is a list of 5W1H properties instantiated on episode part `SocialEpisodePart_2a` and their corresponding values.

?whtag	?property5W1H	?value
<epmem:hasHowTag>	<epmem:How_2a>	"played"@en
<epmem:hasWhoTag>	<epmem:Who_2a>	"I"@en
<epmem:hasWhenTag>	<epmem:When_2a>	"today"@en
<epmem:hasSentimentTag>	<epmem:Sentiment_2a>	"+"^^<xsd:string>
<epmem:hasWhatTag>	<epmem:What_2a>	"sports"@en
<epmem:hasWhereTag>	<epmem:Where_2a>	"at school"@en

5.4.6 Design criterion 4b

To verify if the ontology is able to detect contradictions from similar episodes, a fifth episode was created in the same way the first one was. However, this time the episode was created following the third example, which was given in Section 3.3.3 on page 38. This means the 5W1H properties now are different from the first instantiation.

A SPARQL query was constructed to find all episodes similar to this fifth, but with at least one property contradicting that of the fifth episode. This query differs only slightly from the previous SPARQL query that was given. The results for this query are shown in Table 5.7. In this table, each of the contradicting property values for similar episodes is shown next to the value for the last episode. The last column shows the number of 5W1H property values for that episode part that was the same as on the most recent episode. The SPARQL query itself can be found in Listing B.7.

Table 5.7: The result of a query to find contradicting values. The result is a set of 5W1H properties – and their values – of episodes different from `SocialEpisode_5`, which are the same on the basis of at least two other 5W1H properties.

similarEpisode	whtag	property5W1H	value	contradicts	n
SocialEpisodePart_4a	hasWhatTag	What_5	sports	everything	2
SocialEpisodePart_2a	hasWhatTag	What_5	sports	everything	2
SocialEpisodePart_2a	hasSentimentTag	Sentiment_5	+	-	2
SocialEpisodePart_3a	hasSentimentTag	Sentiment_5	+	-	2
SocialEpisodePart_3a	hasWhatTag	What_5	sports	everything	2
SocialEpisodePart_1a	hasWhatTag	What_5	sports	everything	2

The approach for the HFC database is the same as in Example 2, since the similarities and difference have to be counted in the code behind the query and not in the HFC query itself.

5.5 Design criterion 5

The method for verifying the fifth design criterion was given in example 4 in Section 3.3.3 on page 39. Using a single query, the emotional state associated with an episode, the mood and emotion associated with that state, and their corresponding values and labels can be found. The result of this query is given in Table 5.8. The full query can be found in Listing B.8.

Table 5.8: The result of a SPARQL query requesting the emotional states associated with `epmem:SocialEpisode_3`.

episodePart	SocialEpisodePart_3b	SocialEpisodePart_3a
EmotionalState	EmotionalState_x	EmotionalState_x
emotion	Emotion_a	Emotion_a
emotionLabel		
emotionIntensity	118	118
emotionValence	-134	-134
mood	Mood_y	Mood_y
moodIntensity	-112	-112
moodValence	112	112
moodLabel		
response	Response_q	Response_q

Since the mock-up data from Sternheim (2017) was used, where the label was not set for the given mood and emotion, the label is lacking in the results. In this mock-up example, both episode parts of `epmem:SocialEpisode_3` were associated with the same emotional state. In practice, this is not guaranteed, and different emotional states could be the result of the same query.

In the result set, an instance `Response_q` can be seen. This is a member of the ontology modeling emotional state and the product of a specific instantiation of mood and emotion. It indicate to the PAL system in what way to provide emotional support to the child. Sternheim (2017) suggested four possible types of response, one for each possible valence sign of the mood and emotion. For example, when the current emotion of the child has a negative valence, but the longer lasting (and positively influencing) mood has a positive valence, the PAL actor can be empathetic as well as supportive, by saying: "I'm sorry you feel that way, but I'm sure you'll feel better soon."

A similar query can be constructed in HFC. The emotion and mood label have intentionally been left out of this query, so the result set would not be empty. As can be seen in the results of this query in Table 5.9, the result is the sameas that of Table 5.8, except for these missing labels.

These results show design criterion 5 is satisfied by the ORACLE, since the ESMAS ontology is now well integrated with the other PAL ontologies.

Table 5.9: The result of an HFC query requesting the emotional states associated with `epmem:SocialEpisode_3`.

?episodePart	<epmem:SocialEpisodePart_3a>	<epmem:SocialEpisodePart_3b>
?emotionalState	<epmem:EmotionalState_x>	<epmem:EmotionalState_x>
?emotion	<affect:Emotion_a>	<affect:Emotion_a>
?emotionValence	"-134"^^<xsd:int>	"-134"^^<xsd:int>
?emotionIntensity	"118"^^<xsd:int>	"118"^^<xsd:int>
?mood	<affect:Mood_y>	<affect:Mood_y>
?moodValence	"112"^^<xsd:int>	"112"^^<xsd:int>
?moodIntensity	"-112"^^<xsd:int>	"-112"^^<xsd:int>
?response	<affect:Response_q>	<affect:Response_q>

5.6 Design criterion 6

The sixth and last design criterion stated the unaltered utterances of the child should be stored in such a way they can be used to build natural language corpora and can be traced back from any episode. The way user input is stored in the ORACLE was discussed in Section 4.2 on page 44. A user utterance is split into multiple sentences. These sentences are the input for the sentiment mining algorithm and have the resulting sentiment associated with them. Each of these sentences is then further split into sentence parts, which correspond with one of the seven 5W1H properties on an episode part. This episode part is itself associated with an episode. Following this path, the episode that encodes a user utterance can be found, but traversing this path in the opposite direction allows the unaltered source of the episode to be extracted from the ontology as well.

In Section 4.2.2 this behavior was further extended, to distinguish user utterances that are a response to a specific question from other utterances of the user and to relate these answers to the episodes the questions were asked about. Using this approach, the original question posed by the PAL actor can be related to the given answer as well. This allows building a corpus not only of all utterances of any user in the system, but extending that corpus with examples of how specific questions are answered by the children as well.

This final design criterion is met by the ORACLE as well.

Chapter 6
Conclusion and Discussion

In this chapter, the results of this research will be discussed and the research questions will be answered. After that, some limitations and suggestions for further research will be given. The chapter ends with a conclusion to this research.

6.1 Discussion

The ORACLE was developed to encode events from a user's life in such a manner that questions can be formulated about those events. From the literature it became clear that questions are usually asked because of missing or contradicting information in an internal knowledge base. For that reason, a knowledge base was used as the basis for representing events. By encoding an event using a 5W1H structure, missing information can be easily identified, and the type of information that is missing can be used to suggest a question. By inferring missing values from similar previous episodes, the system cannot only determine when to ask a question, but also when *not* to ask a question.

The final result builds on the counseling agent of Han et al. (2013), as it uses the same 5W1H structure to determine a question to ask in response to a user utterance. However, it extends that approach significantly with the concept of memory. This extension allows the system not only to react to new information statically, but also place that new information in context of past information and reason over that information.

Related work on episodic memory in intelligent agents exists in abundance. However, no related work has been found that provides the advantages to research such as the PAL project as the current research results do. Episodic memory is often used to learn how actions affect the environment of an agent (Nuxoll and Laird, 2004, 2007; Gorski and Laird, 2011; Anderson, 2015). When episodic memory is added to social robots, this is often only aimed at small talk to keep a human talking, rather than to convey actual interest (Bang et al., 2015; Han

et al., 2015). If motivation is a goal of the social robot, the memory system is often designed for only a few sessions with the user, sometimes as little as two (Kasap and Magnenat-Thalmann, 2012) or as a companion for life (de Campos, 2010), neither of which generalizes well to users who interact with the robot for three to four months, such as in the PAL project. Lastly, no related work was found that acknowledges the potential value of the raw, unprocessed user utterances for self-management E-health systems, while a case has been made in the current research that this data is valuable both to a treating health care professional to monitor progress, as for future research as training data.

In the remainder of this section, the research questions formulated in the first chapter will be discussed.

The first subquestion for this research was what types of episodes are useful for the PAL actor. With the originally envisioned approach, question formulating would only be possible with a rich enough understanding of the content of an episode. This question was therefor initially intended to restrict the types of episodes to those types that could be understood with only the domain knowledge available in the PAL system. With direction of the final approach, however, this question can simply be answered with "any episode from which 5W1H information can at least partially be extracted." The ORACLE is in principle not restricted to a limited domain, as it is less dependent on contextual understanding because questions are based on structure, not content, and because it allows using knowledge from external knowledge bases.

The second subquestion was how the PAL system can work with episodes with incomplete annotation. Initially, the view was that annotation should come from the context of the application, but since the episodic memory has been modeled as a knowledge base that is constantly to be extended through question asking, the answer to this subquestion becomes "by asking questions about the missing annotation properties."

The third subquestion was how a follow-up question can be asked after an answer is given. The ORACLE treats an answer to a question as a special kind of user utterance. As long as the answer is a wh-inform statement and can be split into the 5W1H structure, a follow-up question can be formulated in exactly the same way as any of the other questions suggested by this model is formulated.

The fourth and last subquestion was how the existing ontology should be extended or adapted. The existing ontology did not incorporate any of the encoding mechanisms required for representing and reasoning over natural language input. The existing ontology had to be extended with the entire ontology implementation of the ORACLE model. Furthermore, the ontology for affect, representing the ESMAS, had to be incorporated within the `PAL` ontology to allow reasoning over the emotional state of the child.

Together, these subquestions help to answer the main question for this research:

> "How can episodes and sentiments in the PAL system be used to ask personalized open questions?"

The answer to this question is that episodes can be regarded as information about events in the child's life. The PAL system can use this information to formulate questions about information that can be determined to be absent in the episode. Sentiment can be regarded as information that may be present or absent, but can also be reflected in the formulation of a question the system asks about that event. However, as became clear from the literature, sentiment alone is not sufficient to describe the emotional state of a child, so to maximize the personalization, mood and emotion also need to be taken into account when using affect to ask a personalized open question.

6.2 Limitations and suggestions for further research

Although the current research was conducted with care it is not without defects. First and foremost, the current research has focused entirely on the encoding of episodes in an ontology and the mechanisms required for reasoning over those episodes. However, before this encoding and reasoning can be used, a sufficiently capable parsing algorithm is required, which can split user utterances into sentences or subordinate clauses, so each sentence or subordinate clause concerns just one topic. Further more, that algorithm needs to be able to, at the very least, extract 5W1H properties from those sentences and subordinate clauses. Without such an algorithm, no content is available to the ORACLE to reason over at all. Although Han et al. (2013) suggest an approach for extracting these 5W1H properties, even they acknowledge this approach has only been shown to work in Korean text. If their approach for extracting these properties from natural language generalizes well to other languages is a matter for future research.

That is not to say the approach of Han et al. (2013) is the only approach that could work. There are many natural language understanding (NLU) algorithms available, and the number keeps growing. One approach, which seems viable, is the Rasa NLU toolkit[1] (Bocklisch et al., 2017). This open source module for natural language understanding and dialog management is able to learn user intentions topics of relevance of utterances with relatively little training data, and current work is ongoing to integrate this toolkit for similar purposes in the PAL system. Future research is needed to determine what algorithm can be used for parsing

[1] https://rasa.com/

natural language utterances into the required 5W1H properties, especially when dealing with child-generated natural language.

The lack of dialog manager is another limitation in this research. The dialog manager currently incorporated in the PAL system uses predefined scripts, which are fired in very specific circumstances (e.g. greeting, ask quiz question, give feedback on information entered in time line, ask a specific question about a specific episode). It is unclear whether scripts can be devised which can translate a suggestion for a type of question from the ORACLE to a proper question. It remains an open question if this dialog manager can be used, or requires updates or even replacement in order to deal with this problem.

Both the lack of dialog manager and that of the natural language understanding required to instantiate the ORACLE form another limitation to the results of this research: The reasoning aspect of the ORACLE has been verified using queries. However, a full user test in a natural context has not been performed. It is therefor impossible to say conclusively that the model will handle unexpected user input as well as the given examples and if the proposed questions do indeed increase relatedness between the user and the PAL actor. Once the ORACLE has been integrated with tools for natural language understanding and dialog management capable of parsing user input correctly and formulating questions on the basis of suggestions by the ORACLE, further research is required to validate this claim.

When example utterances were provided in Section 4.1.1 on page 41, it already became clear there is no unique way of instantiation a 5W1H model for every given user utterance. Ambiguity exists for which parts of the utterance correspond with what 5W1H properties. Additional research is required to determine if there is a clear way to settle this ambiguity, or if this ambiguity can be dealt with in a more consistent way than has been done in this research.

Another limitation lies with the original aim of this research. The project started out to investigate how the PAL actor could ask open questions, and how to ask a follow-up question after an answer was given, under the assumption that this would further personalize the PAL actor and increase relatedness with the child. However, according to Kearsley (1976), chaining of open questions is often used to assert dominance. The aim of the PAL project is to have Charlie become a pal of the children, on the same level in the social hierarchy. Kearsley suggests embedded and yes/no questions work to enable this goal. Further research is advised to understand how questions can be used to ensure equality between a social robot and children in the ages of 8 to 14 years. There is, however, a more simply learned lesson for the PAL project. Currently, children can only enter open text as a description of a diary event, or as an answer to the robot. For an equal status in the social hierarchy, it is advisable for the PAL project to allow children to initiate dialog when they want, in addition to allowing children to answer questions

of the PAL actor.

Even with these limitations, however, the current research provides a valuable addition to the PAL project. First of all, while the sentiment mining algorithm is in place and active in the PAL project, there is not yet a method for storing it in the HFC database. Even if the episodes in the ORACLE are not instantiated, this model allows storing all utterances of the user and PAL actor in the database and, more importantly, to encode those utterances with sentiment whenever available. Lastly, ORACLE allows to model how an utterance of the child is caused by one of the PAL actor's and vice versa. By starting to use the model in this way, the generation of a child-generated corpus can start before the episodic memory part of this model is taken in use. This corpus can then be used to advance the state of the art of various natural language tools.

Further more, the current research efforts have resulted in an integration of the ESMAS into the all-encompassing `PAL` ontology, which means an emotional state could now be instantiated for any agent in the PAL domain. This emotional state could be used to increase the empathy of the robot towards the child, but also to model the PAL agent as having an emotional state, possibly further increasing its believability as a "pal" for the children. Further research is required to investigate how the emotional state of a child can be determined from their behavior.

Lastly, one of the requirements defined in Chapter 3 on page 28 was that the model be easily portable to other domains or even completely other systems. Provided these systems make use of an ontology, this is possible by taking the ORACLE ontology, without any of the other ontologies used within the PAL project. This ontology could then be used not only for personalized open question asking, but might also be useful for dialog agents that have the task of informing, rather than asking. By the same principles the ORACLE suggests a question for a wh-property missing on an episode, an episode can be instantiated as encoding a fact, and a wh-inform statement could be constructed using the 5W1H properties that are present, rather than absent. This approach could be viable for efforts currently ongoing in the field of explainable AI, which seeks ways decisions made by black-box AI algorithms could be explained to a user.

6.3 Conclusion

In this research, the ORACLE (Ontology Reasoner for Affective Conversation in Long-term Interaction) model was presented, which allows encoding of informative statements in a clear structure, reasoning over those representations, and forming questions to extend or correct those representations. This model was developed as a way to allow a social robot to ask personalized open questions about events from the user's life, with the PAL project – aimed at children with diabetes between the

ages of 8 and 14 years – as the specific use case, to increase relatedness between those children and the PAL actor.

Although the initial research direction was to extend the functionality of the Episodic Memory Manager developed within the context of the PAL project, the literature suggested an approach where dialog management was assumed to exist and where focus lay on the encoding and reasoning mechanisms required.

For the design of the ORACLE, theories from question asking and episodic memory were combined with ideas from the field of social robotics and counseling agents. The ORACLE has been integrated within other ontologies used by the PAL project.

The ORACLE is able to reason over events relayed to it by natural language by splitting utterances into a 5W1H structure, where sentiment is included. The ORACLE model detects an opening for a question in one of two ways. Firstly, when a user tells the social robot about an event, the ORACLE is able to determine what type of information is underrepresented, and can suggest a question on that topic. For the second way, the ORACLE makes use of knowledge from past events that are similar to a new event. If past events contradict newly acquired information, the ORACLE can suggest a question on why or how this new event differs from the old one.

The ORACLE distinguishes itself from related work in a few ways. First of all, it does not use episodic memory to learn how actions affect it's environment, but rather to relate and reason over various events from a user's life. Where other social agent's may use episodic memory for small talk, the ORACLE is aimed at asking meaningful, personalized questions about events from the user's life, not only for the sake of conversation, but to learn new information that may be useful for future events as well. Further more, most related work where episodic memory is combined with a social agent either focuses on a very limited number of interactions, or on becoming a life-long companion. The ORACLE is specifically aimed at user's using the system for longer periods of time, but not indefinitely, focusing on single events rather than entire life time periods or, conversely, small details of an interaction. Lastly, the ORACLE supports using affect and sentiment of a user not only to determine what question to ask, but how to formulate that question in an empathetic way. Lastly, the ORACLE provides a way to the PAL project to not only model episodic memory, but build a data base of child generated natural language utterances, combined with sentiment as well.

The research in this project is limited by its focus on the encoding of and reasoning over episodic events, rather than an integrated approach including parsing of natural language and dialog management. This means no parsing algorithms or dialog manager have been shown to work, nor has it been determined that the questions the ORACLE allows a dialog manager to ask positively affect a user's

bond with the social robot. Further research is suggested for the design of proper parsing algorithms and dialog management, as well as to validate the effect question asking using the ORACLE has on users of E-Health applications or social robots.

Bibliography

Anderson, T. S. (2015). From episodic memory to narrative in a cognitive architecture. In *OASIcs-OpenAccess Series in Informatics*, volume 45. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Bang, J., Han, S., Lee, K., and Lee, G. G. (2015). Open-domain personalized dialog system using user-interested topics in system responses. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 771–776. IEEE.

Baxter, P., Wood, R., and Belpaeme, T. (2012). A touchscreen-based'sandtray'to facilitate, mediate and contextualise human-robot social interaction. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 105–106. ACM.

Bechhofer, S. (2009). Owl: Web ontology language. In *Encyclopedia of database systems*, pages 2008–2009. Springer.

Belpaeme, T., Baxter, P. E., Read, R., Wood, R., Cuayáhuitl, H., Kiefer, B., Racioppa, S., Kruijff-Korbayová, I., Athanasopoulos, G., Enescu, V., et al. (2012). Multimodal child-robot interaction: Building social bonds. *Journal of Human-Robot Interaction*, 1(2):33–53.

Bocklisch, T., Faulker, J., Pawlowski, N., and Nichol, A. (2017). Rasa: Open source language understanding and dialogue management. *arXiv preprint arXiv:1712.05181*.

Brom, C., Lukavskỳ, J., and Kadlec, R. (2010). Episodic memory for human-like agents and human-like agents for episodic memory. *International Journal of Machine Consciousness*, 2(02):227–244.

Castellano, G., Aylett, R., Dautenhahn, K., Paiva, A., McOwan, P. W., and Ho, S. (2008). Long-term affect sensitive and socially interactive companions. In *Proceedings of the 4th International Workshop on Human-Computer Conversation*.

Castensøe-Seidenfaden, P., Teilmann, G., Kensing, F., Hommel, E., Olsen, B. S., and Husted, G. R. (2017). Isolated thoughts and feelings and unsolved concerns: adolescents' and parents' perspectives on living with type 1 diabetes–a qualitative study using visual storytelling. *Journal of clinical nursing*, 26(19-20):3018–3030.

de Campos, J. C. F. (2010). May: my memories are yours an interactive companion that saves the user's memories.

Deci, E. L. and Ryan, R. M. (1985). The general causality orientations scale: Self-determination in personality. *Journal of research in personality*, 19(2):109–134.

Duffy, B. R. (2003). Anthropomorphism and the social robot. *Robotics and autonomous systems*, 42(3-4):177–190.

Fumagalli, M., Bierman, B., Kiefer, B., Broekens, J., Demiris, Y., and Neerincx, M. Dr 5.1: Pal technical architecture and software architecture.

Gorski, N. A. and Laird, J. E. (2011). Learning to use episodic memory. *Cognitive systems research*, 12(2):144–153.

Hahn, U. and Romacker, M. (1999). Syndikate-generating text knowledge bases from natural language texts. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, volume 5, pages 918–923. IEEE.

Han, S., Bang, J., Ryu, S., and Lee, G. G. (2015). Exploiting knowledge base to generate responses for natural language dialog listening agents. In *SIGDIAL Conference*, pages 129–133.

Han, S., Lee, K., Lee, D., and Lee, G. G. (2013). Counseling dialog system with 5w1h extraction. In *SIGDIAL Conference*, pages 349–353.

Henkemans, O. A. B., Bierman, B. P., Janssen, J., Looije, R., Neerincx, M. A., van Dooren, M. M., de Vries, J. L., van der Burg, G. J., and Huisman, S. D. (2017). Design and evaluation of a personal robot playing a self-management education game with children with diabetes type 1. *International Journal of Human-Computer Studies*.

Ho, W. C., Dautenhahn, K., Lim, M. Y., Vargas, P. A., Aylett, R., and Enz, S. (2009). An initial memory model for virtual and robot companions supporting migration and long-term interaction. In *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*, pages 277–284. IEEE.

Hovy, E. H. (2015). What are sentiment, affect, and emotion? applying the methodology of michael zock to sentiment analysis. In *Language production, cognition, and the Lexicon*, pages 13–24. Springer.

International Diabetes Federation (2017). IDF Diabetes Atlas. http://www.diabetesatlas.org.

Jockel, S., Weser, M., Westhoff, D., and Zhang, J. (2008). Towards an episodic memory for cognitive robots. In *Proc. of 6th Cognitive Robotics workshop at 18th European Conf. on Artificial Intelligence (ECAI)*, pages 68–74. Citeseer.

Kasap, Z. and Magnenat-Thalmann, N. (2010). Towards episodic memory-based long-term affective interaction with a human-like robot. In *RO-MAN, 2010 IEEE*, pages 452–457. IEEE.

Kasap, Z. and Magnenat-Thalmann, N. (2012). Building long-term relationships with virtual and robotic characters: the role of remembering. *The Visual Computer*, 28(1):87–97.

Kearsley, G. P. (1976). Questions and question asking in verbal discourse: A cross-disciplinary review. *Journal of psycholinguistic research*, 5(4):355–375.

Kiefer, B. (2012). The talking robots toolkit: Documentation for the content planning module. Technical report, DFKI GmbH, Saarbrücken, Germany. Retrieved: http://talkingrobots.dfki.de/tarot.

Klyne, G. and Carroll, J. J. (2006). Resource description framework (rdf): Concepts and abstract syntax.

Krieger, H.-U. (2012). A temporal extension of the hayes/ter horst entailment rules and an alternative to w3c's n-ary relations. In *FOIS*, pages 323–335.

Krieger, H.-U. (2016). Integrating graded knowledge and temporal change in a modal fragment of owl. In *International Conference on Agents and Artificial Intelligence*, pages 75–95. Springer.

Krieger, H.-U., Peters, R., Kiefer, B., van Bekkum, M. A., Kaptein, F., and Neerincx, M. A. (2016). The federated ontology of the pal project. interfacing ontologies and integrating time-dependent data. In *8th International Joint Conference on Knowledge Engineering and Ontology Development*. INSTICC, SCITEPRESS.

Kruijff-Korbayová, I., Oleari, E., Bagherzadhalimi, A., Sacchitelli, F., Kiefer, B., Racioppa, S., Pozzi, C., and Sanna, A. (2015). Young users' perception of a social robot displaying familiarity and eliciting disclosure. In *International Conference on Social Robotics*, pages 380–389. Springer.

Ligthart, M. (2016). Mypal: A digital diabetes diary with a responsive social avatar.

Looije, R. (2015). Pal for diabetic children. https://www.tno.nl/en/focus-areas/healthy-living/predictive-health-technologies/pal-for-diabetic-children/. Retrieved: 2017-12-07.

Looije, R., Neerincx, M. A., Peters, J. K., and Henkemans, O. A. B. (2016). Integrating robot support functions into varied activities at returning hospital visits. *International Journal of Social Robotics*, 8(4):483–497.

Nuxoll, A. and Laird, J. E. (2004). A cognitive model of episodic memory integrated with a general cognitive architecture. In *ICCM*, pages 220–225. Citeseer.

Nuxoll, A. M. and Laird, J. E. (2007). Extending cognitive architecture with episodic memory. *Ann Arbor*, 1001:48109–2121.

Nuxoll, A. M. and Laird, J. E. (2012). Enhancing intelligent agents with episodic memory. *Cognitive Systems Research*, 17:34–48.

Peters, R., Broekens, J., and Neerincx, M. A. (2017). Guidelines for tree-based collaborative goal setting. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, pages 401–405. ACM.

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., and Wroe, C. (2004). Owl pizzas: Practical experience of teaching owl-dl: Common errors & common patterns. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 63–81. Springer.

Ryan, R. M. and Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American psychologist*, 55(1):68.

Schreuder Goedheijt, B. (2017). Recalling shared memories in an embodied conversational agent. Master's thesis, KTH Information and Communication Technology.

Sternheim, A. (2017). Affective semantics: An ontology for affective reasoning in the pal project. unpublished research internship report, available on request at TNO (Dutch Organisation for Applied Sciences).

Tulving, E. (2002). Episodic memory: From mind to brain. *Annual review of psychology*, 53(1):1–25.

Tulving, E. et al. (1972). Episodic and semantic memory. *Organization of memory*, 1:381–403.

Van Stee, C. (2017). Pal feedback to increase motivation of children with diabetes during mypal usage.

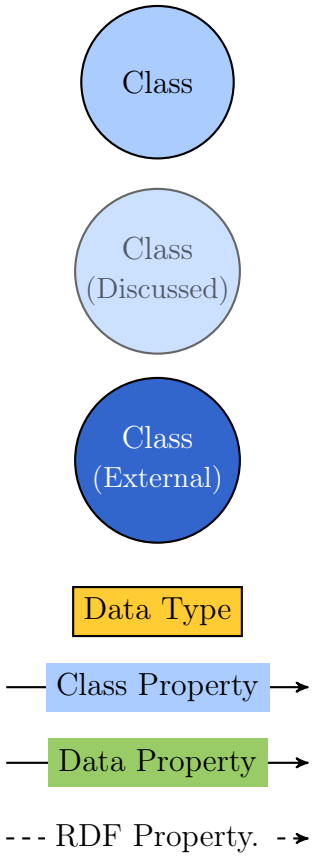
Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

World Health Organization (2017). Diabetes fact sheet no. 312. http://www.who.int/mediacentre/factsheets/fs312/en/.

Appendices

Appendix A

Legend for Visualized Ontologies



Class that already occured in one of the previous figures. Listed properties may be incomplete.

Placeholder class that is defined in other ontology. Will be linked to that other class by equivalence relation.

Data types do not have names. Instead, the label in the box indicates what type of data is stored.

If no label is present, this indicates a subclass relation.

Figure A.1: Legend for figures visualizing the ontology.

Appendix B
Queries

B.1 Queries for Design Criterion 4

B.1.1 SPARQL

```
SELECT
        ?eppart ?tag ?source ?cue
where {
        epmem:SocialEpisode_1 epmem:hasEpisodePart ?eppart .
        ?eppart ?whTag ?tag .
        ?tag a ?t .
        ?t rdfs:subClassOf epmem:Property5W1H .
        optional {
                ?tag epmem:hasSource ?source
        } .
        optional {
                ?tag epmem:hasEpisodeCue ?cue
        }
}
```

Listing B.1: A SPARQL query to find all 5W1H properties for all episode parts of episode epmem:SocialEpisode_1, including their source. The OPTIONAL operator assures properties which do not yet have a value are returned the same as those that do. This query will be generated in the code behind.

B.1.2 HFC

```
select
        ?episodePart ?t1 ?tag ?t2 ?source ?content
where
        <epmem:SocialEpisode_1> <epmem:hasEpisodePart>
        ?episodePart ?t1 &
        ?episodePart <epmem:hasWHTag> ?tag ?t2 &
        ?tag <epmem:hasSource> ?source ?t3 &
        ?source <epmem:hasContent> ?content ?t4
```

Listing B.2: A query to the HFC database to find all property sources for all episode parts of a given episode (in this instance, `SocialEpisode_1`)

```
select
        ?tag ?cue
where
        <epmem:SocialEpisodePart_1a> <epmem:hasWHTag> ?tag ?_ &
        ?tag <epmem:hasEpisodeCue> ?cue ?_
        filter
        ?tag != <epmem:How_1a> &
        ?tag != <epmem:Who_1a> &
        ?tag != <epmem:When_1a> &
        ?tag != <epmem:Where_1a> &
        ?tag != <epmem:What_1a>
```

Listing B.3: A query find all tags that were not found by the HFC query given in Listing B.2. This query will be generated in the code behind.

B.2 Queries Design Criterion 4a

B.2.1 SPARQL

```
SELECT
        ?similarEpisode ?whtag ?property5W1h ?value ?nSimilarTags
where {
        {
                select
                        (sample(?otherEpisodePart)
                                AS ?similarEpisode)
                        (count(?otherTag) AS ?nSimilarTags)
                where {
                        epmem:SocialEpisode_4
                                epmem:sharedWith ?child .
                        epmem:SocialEpisode_4
                                epmem:hasEpisodePart ?episodePart .
                        ?episodePart ?hasWHTag ?tag .
                        ?tag a ?type .
                        ?type rdfs:subClassOf epmem:Property5W1H .
                        ?tag epmem:hasSource ?source .
                        ?source epmem:hasContent ?content .
                        ?otherSource epmem:hasContent ?otherContent .
                        FILTER (?otherSource != ?source )
                        FILTER(?content = ?otherContent)
                        ?otherTag epmem:hasSource ?otherSource .
                        ?otherEpisodePart ?hasWHTag ?otherTag .
                        ?otherEpisode
                                epmem:hasEpisodePart ?episodePart .
                        ?otherEpisode epmem:sharedWith ?child
                }
                GROUP BY ?otherEpisodePart
                HAVING (?nSimilarTags > 2)
        }

        ?similarEpisode ?whtag ?property5W1h .
        ?property5W1h a ?type .
        ?type rdfs:subClassOf epmem:Property5W1H .
        ?property5W1h epmem:hasSource ?source
        FILTER NOT EXISTS {
                epmem:SocialEpisode_4
```

```
                        epmem:hasEpisodePart ?episodePart .
                ?episodePart ?whtag ?tag .
                ?tag epmem:hasSource ?_
        } .
        ?source epmem:hasContent ?value
}
```

Listing B.4: A SPARQL query that finds all all 5W1H property values which are missing on the current episode, but are present on another episode which is similar on at least 3 other 5W1H properties.

B.2.2 HFC

```
select distinct
        ?similarEpisodePart
        ?whtag
        ?property5W1H
        ?value
where
        <epmem:SocialEpisode_4> <epmem:sharedWith> ?child ?_ &
        <epmem:SocialEpisode_4>
                <epmem:hasEpisodePart> ?episodePart ?_ &
        ?episodePart ?whtag ?tag ?_ &
        ?tag <rdf:type> <epmem:Property5W1H> ?_ &
        ?tag <epmem:hasSource> ?source ?_ &
        ?source <epmem:hasContent> ?value ?_ &
        ?otherSource <epmem:hasContent> ?value ?_  &
        ?property5W1H <epmem:hasSource> ?otherSource ?_ &
        ?similarEpisodePart ?whtag ?property5W1H ?_
filter
        ?whtag != <epmem:hasWHTag> &
        ?source != ?otherSource
```

Listing B.5: The first HFC query finds all 5W1H properties that are similar to those of the current episode, and the episode parts they belong to.

```
select
        ?whtag ?property5W1H ?value
where
        <epmem:SocialEpisodePart_2a> ?whtag ?property5W1H ?_ &
        ?property5W1H <epmem:hasSource> ?source ?_ &
        ?source <epmem:hasContent> ?value ?_
filter
        ?whtag != <epmem:hasWHTag>
```

Listing B.6: For each episode part that has sufficient similar properties to the episode it is compared to, all property values are requested. The properties that are not instantiated on the episode it is compared to could be inferred.

B.3 Queries for Design Criterion 4b

```
SELECT
        ?similarEpisode
        ?whtag
        ?property5W1H
        ?value
        ?contradicts
        ?nSimilarTags
where {
        {
        select
                (sample(?otherEpisodePart) AS ?similarEpisode)
                (count(?otherTag) as ?n)
        where {
                epmem:SocialEpisode_5 epmem:sharedWith ?child .
                epmem:SocialEpisode_5
                        epmem:hasEpisodePart ?episodePart .
                ?episodePart ?hasWHTag ?property5W1H .
                ?property5W1H a ?type .
                ?type rdfs:subClassOf epmem:Property5W1H .
                ?property5W1H epmem:hasSource ?source .
                ?source epmem:hasContent ?content .
                ?otherSource epmem:hasContent ?otherContent .
                FILTER (?otherSource != ?source)
                FILTER (?otherContent = ?content)
                ?otherTag epmem:hasSource ?otherSource .
                ?otherEpisodePart ?hasWHTag ?otherTag .
                ?otherEpisode
                        epmem:hasEpisodePart ?otherEpisodePart .
                ?otherEpisode epmem:sharedWith ?child
                }
        GROUP BY ?otherEpisodePart
        HAVING (?n > 1)
        }
        ?similarEpisode ?whtag ?property .
        ?property a ?type .
        ?type rdfs:subClassOf epmem:Property5W1H .
        ?property epmem:hasSource ?source .
        ?source epmem:hasContent ?value .
        epmem:SocialEpisode_5 epmem:hasEpisodePart ?episodePart .
```

```
        ?episodePart ?whtag ?property5W1H .
        optional {
                ?property5W1H epmem:hasSource ?oursource .
                ?oursource epmem:hasContent ?contradicts
        }
        filter (?contradicts != ?value)
}
```

Listing B.7: A SPARQL query that finds all all 5W1H property values on another episode which is similar on at least 2 other 5W1H properties, but which are contradicting those of the current episode.

B.4 Queries for Design Criterion 5

B.4.1 SPARQL

```
SELECT
        ?episodePart ?EmotionalState ?emotion
        ?emotionLabel ?emotionIntensity
        ?emotionValence ?mood
        ?moodIntensity ?moodValence
        ?moodLabel ?response
WHERE {
        epmem:SocialEpisode_3 epmem:hasEpisodePart ?episodePart .
        ?episodePart epmem:occuredAtEmotionalState
                ?emotionalStatePlaceholder .
        ?emotionalStatePlaceholder owl:sameAs ?EmotionalState .
        ?EmotionalState affect:containsEmotion ?emotion .
        ?EmotionalState affect:containsMood ?mood .
        OPTIONAL {
                ?mood affect:hasLabel ?moodLabel .
        }
        OPTIONAL {
                ?mood affect:hasIntensityValue ?moodIntensity .
                ?mood affect:hasValenceValue ?moodValence .
        }
        OPTIONAL {
                ?emotion affect:hasLabel ?emotionLabel .
        }
        OPTIONAL {
                ?emotion affect:hasIntensityValue ?emotionIntensity .
                ?emotion affect:hasValenceValue ?emotionValence .
        }
        ?EmotionalState affect:causesResponse ?response
}
```

Listing B.8: Using a single query, concepts of the Episodic Memory ontology can be linked to those of the ESMAS ontology, since both ontologies have now been integrated within the all-encompassing PAL ontology. This query finds the emotional state – and the therein enclosed mood and emotion – that was active at the time SocialEpisode_3 was inserted into the database

B.4.2 HFC

```
select
        ?episodePart
        ?emotionalState
        ?emotion
        ?emotionValence
        ?emotionIntensity
        ?mood
        ?moodValence
        ?moodIntensity
        ?response
where
        <epmem:SocialEpisode_3>
        <epmem:hasEpisodePart> ?episodePart ?_ &
        ?episodePart <epmem:occuredAtEmotionalState>
                ?emotionalState ?_ &
        ?emotionalState <affect:containsEmotion> ?emotion ?_ &
        ?emotionalState <affect:containsMood> ?mood ?_ &
        ?emotionalState <affect:causesResponse> ?response ?_ &
        ?emotion <affect:hasIntensityValue> ?emotionIntensity ?_ &
        ?emotion <affect:hasValenceValue> ?emotionValence ?_ &
        ?mood <affect:hasIntensityValue> ?moodIntensity ?_ &
        ?mood <affect:hasValenceValue> ?moodValence ?_
```

Listing B.9: Within the HFC query languages, one query can be constructed to query both the ORACLE and the ESMAS ontology about relating class concepts.