CASAM

FP7-217061

Computer-Aided Semantic Annotation of Multimedia

Deliverable D3.3

Probabilistic abduction engine: Report on algorithms and the optimization techniques used in the implementation

Editor(s):	Oliver Gries, Ralf Möller, Anahita Nafissi, Maurice Rosenfeld, Kamil Sokolski, Michael Wessel
Responsible Partner:	TUHH
Status-Version:	V.2 final
Date:	14/05/2010
EC Distribution:	

Project Number:	FP7-217061
Project Title:	CASAM

Title of Deliverable:	Probabilistic abduction engine: Report on algorithms and the optimization techniques used in the implementation
Date of Delivery to the EC:	

Workpackage responsible for the Deliverable:	WP3 Knowledge representation and reasoning for multimedia interpretation
Editor(s):	Oliver Gries, Ralf Möller, Anahita Nafissi, Maurice Rosenfeld, Kamil Sokolski, Michael Wessel
Contributor(s):	Oliver Gries, Ralf Möller, Anahita Nafissi, Maurice Rosenfeld, Kamil Sokolski, Michael Wessel
Reviewer(s):	Giorgos Apostolikas, Robert Hendley
Approved by:	All partners

Abstract:	This report introduces the probabilistic abduction engine for multimedia interpretation. The architecture of the engine is described and the functionality of its components is discussed. Optimization techniques are briefly explained. An algorithm for performing the interpretation process is proposed. Finally, it is explained how the Abox difference process is computed.
Keyword List:	Probability, Description logic, Markov logic, Abduction, Abox difference

Document Description

Document Revision History

Version	*Date*	*Modifications Introduced*	
		Modification Reason	*Modified by*
V.1	15/04/10	For internal review	Oliver Gries, Ralf Möller, Anahita Nafissi, Maurice Rosenfeld, Kamil Sokolski, Michael Wessel
V.2	14/05/10	For external review	Oliver Gries, Ralf Möller, Anahita Nafissi, Maurice Rosenfeld, Kamil Sokolski, Michael Wessel

Executive Summary

For multimedia interpretation, a semantically well-founded formalization is required. In accordance with previous work, in CASAM a well-founded abduction-based approach is pursued. Extending previous work, abduction is controlled by probabilistic knowledge, and it is done in terms of first-order logic.

This report describes the probabilistic abduction engine and the optimization techniques for multimedia interpretation. It extends deliverable D3.2 by providing a probabilistic scoring function for ranking interpretation alternatives. Parameters for the CASAM Abduction Engine (CAE) introduced already in D3.2 are now appropriately formalized such that CAE is better integrated into the probabilistic framework. In addition, this deliverable describes how media interpretation services can be provided that work incrementally, i.e., are able to consume new analysis results, or new input from a human annotator, and produce notifications for additional interpretation results or, in some cases, revision descriptions for previous interpretations. Incremental processing is nontrivial and is realized using an Abox difference operator, which is used to interpretation results obtained for extended inputs with one(s) previously obtained such that notifications about additions and revisions can be computed.

Contents

1 Introduction

The goal of the CASAM project is to support manual multimedia annotation processes where high level concept assertions are attached as annotations for multimedia documents. In particular, CASAM investigates shot videos as a source for investigation how manual annotation. A human annotator can describe objects in video shots using short natural language texts or interactive graphical means. The goal is to reduce this work by (i) providing sophisticate automatic analysis of text and video and (2), on top of that, exploiting an additional knowledge-based interpretation process for interpreting these analysis results on a more abstract level. For this, in CASAM, a so-called reasoning-based media interpretation (RMI) system is developed as a first prototype.

For multimedia interpretation, a semantically well-founded formalization is required. In accordance with previous work, in CASAM a well-founded abduction-based approach is pursued. Extending previous work, abduction is controlled by probabilistic knowledge, and it is done in terms of first-order logic.

In Deliverable D3.2, conceptual architecture of RMI was presented, and the functionality of its components was explained. This report describes the probabilistic abduction engine and introduces optimization techniques for multimedia interpretation. It extends deliverable D3.2 by providing a probabilistic scoring function for ranking interpretation alternatives. Parameters for the CASAM Abduction Engine (CAE) introduced already in D3.2 are now appropriately formalized for the first time such that CAE is better integrated into the probabilistic framework. In addition, this deliverable describes how media interpretation services can be provided that work incrementally, i.e., are able to consume new analysis results, or new input from a human annotator, and produce notifications for additional interpretation results or, in some cases, revision descriptions for previous interpretations. Incremental processing is nontrivial and is realized using an Abox difference operator, which is used to interpretation results obtained for extended inputs with one(s) previously obtained such that notifications about additions and revisions can be computed.

Based on a presentation of the most important preliminaries in Chapter 2, the conceptual architecture for the probabilistic abduction engine is introduced in Chapter 3. The abduction and interpretation procedures are discussed in detail. Optimization techniques for the probabilistic abduction engine are pointed out. In Chapter 4, a complete example is given, showing the main approach using intermediate steps. Chapter 5 describes how a specific operation, namely the Abox difference operation, is implemented. The Abox difference operation is required for realizing the incremental notification of client modules.

2 Preliminaries

In this chapter, the most important preliminaries already explained in detail in Deliverable D3.2 are given in order to make this document self-contained.

2.1 Preliminaries on Description Logic

One of the main targets of the CASAM project is to support human annotators during their work in producing elabortate symbolic descriptions for small video shots. Annotations are used for later information retrieval and require a representation language. We assume that a less expressive description logic should be applied to facilitate fast computations. We decided to represent the domain knowledge with the DL $\mathcal{ALH}_f{}^-(\mathcal{D})$ (restricted attributive concept language with role hierarchies, functional roles and concrete domains). We shortly describe our nomenclature in order to make this deliverable self-contained. For details see [Baader et al., 2003].

In logic-based approaches, atomic representation units have to be specified. The atomic representation units are fixed using a so-called signature. Whereas, up to now, the signature is defined manually, in upcoming deliverables, machine learning techniques might be used to extend the signature automatically such that RMI can be adapted to new contexts.

A DL *signature* is a tuple $\mathcal{S} = (\mathbf{CN}, \mathbf{RN}, \mathbf{IN})$, where $\mathbf{CN} = \{A_1, ..., A_n\}$ is the set of concept names (denoting sets of domain objects) and $\mathbf{RN} = \{R_1, ..., R_m\}$ is the set of role names (denoting relations between domain objects). The signature also contains a component $\mathbf{IN}$ indicating a set of individuals (names for domain objects).

In order to relate concept names and role names to each other (terminological knowledge) and to talk about specific individuals (assertional knowledge), a knowledge base has to be specified. An $\mathcal{ALH}_f{}^-$ *knowledge base* $\Sigma_\mathcal{S} = (\mathcal{T}, \mathcal{A})$, defined with respect to a signature $\mathcal{S}$, is comprised of a terminological component $\mathcal{T}$ (called *Tbox*) and an assertional component $\mathcal{A}$ (called *Abox*). In the following we just write Σ if the signature is clear from context. A Tbox is a set of so-called *axioms*, which are restricted to the following form in $\mathcal{ALH}_f{}^-$:

(I)	Subsumption	$A_1 \sqsubseteq A_2$, $R_1 \sqsubseteq R_2$
(II)	Disjointness	$A_1 \sqsubseteq \neg A_2$
(III)	Domain and range restrictions for roles	$\exists R.\top \sqsubseteq A$, $\top \sqsubseteq \forall R.A$
(IV)	Functional restriction on roles	$\top \sqsubseteq (\leq 1\, R)$
(V)	Local range restrictions for roles	$A_1 \sqsubseteq \forall R.A_2$
(VI)	Definitions with value restrictions	$A \equiv A_0 \sqcap \forall R_1.A_1 \sqcap ... \sqcap \forall R_n.A_n$

With axioms of form (I), concept (role) names can be declared to be subconcepts (subroles) of each other. Axioms of form (II) denote disjointness between concepts. Axioms of type (III) introduce domain and range restrictions for roles. Axioms of the form (IV) introduce so-called *functional* restrictions on roles, and axioms of type (V) specify local range restrictions (using value restrictions, see below). With axioms of kind (VI) so-called definitions (with necessary and sufficient conditions) can be specified for concept names found on the lefthand side of the $\equiv$ sign. In the axioms, so-called *concepts* are used. Concepts are concept names or expressions of the form $\top$ (anything), $\bot$ (nothing), $\neg A$ (atomic negation), $(\leq 1\, R)$ (role functionality), $\exists R.\top$ (limited existential restriction), $\forall R.A$ (value restriction) and $(C_1 \sqcap ... \sqcap C_n)$ (concept conjunction).

Knowledge about individuals is represented in the Abox part of Σ. An Abox $\mathcal{A}$ is a set of expressions of the form $A(a)$ or $R(a, b)$ (concept assertions and role assertions, respectively) where A stands for a concept name, R stands for a role name, and a, b stand for individuals. Aboxes can also contain equality $(a = b)$ and inequality assertions $(a \neq b)$. We say that the unique name assumption (UNA) is applied, if $a \neq b$ is added for all pairs of individuals a and b.

In order to understand the notion of logical entailment , we introduce the semantics of $\mathcal{ALH}_f{}^-$. In DLs such as $\mathcal{ALH}_f{}^-$, the semantics is defined with interpretations $\mathcal{I} = (\triangle^\mathcal{I}, \cdot^\mathcal{I})$, where $\triangle^\mathcal{I}$ is a non-empty set of domain objects (called the domain of $\mathcal{I}$) and $\cdot^\mathcal{I}$ is an interpretation function which maps individuals to objects of the domain ($a^\mathcal{I} \in \triangle^\mathcal{I}$), atomic concepts to subsets of the domain ($A^\mathcal{I} \subseteq \triangle^\mathcal{I}$) and roles to subsets of the cartesian product of the domain ($R^\mathcal{I} \subseteq \triangle^\mathcal{I} \times \triangle^\mathcal{I}$). The interpretation of arbitrary $\mathcal{ALH}_f{}^-$ concepts is then defined by extending $\cdot^\mathcal{I}$ to all $\mathcal{ALH}_f{}^-$ concept constructors as follows:

$$\begin{array}{lcl}
\top^{\mathcal{I}} & = & \triangle^{\mathcal{I}} \\
\bot^{\mathcal{I}} & = & \emptyset \\
(\neg A)^{\mathcal{I}} & = & \triangle^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
(\leq 1\, R)^{\mathcal{I}} & = & \{u \in \triangle^{\mathcal{I}} \mid (\forall v_1, v_2)\,[((u, v_1) \in R^{\mathcal{I}} \wedge (u, v_2) \in R^{\mathcal{I}}) \rightarrow v_1 = v_2] \\
(\exists R.\top)^{\mathcal{I}} & = & \{u \in \triangle^{\mathcal{I}} \mid (\exists v)\,[(u, v) \in R^{\mathcal{I}}]\} \\
(\forall R.C)^{\mathcal{I}} & = & \{u \in \triangle^{\mathcal{I}} \mid (\forall v)\,[(u, v) \in R^{\mathcal{I}} \rightarrow v \in C^{\mathcal{I}}]\} \\
(C_1 \sqcap ... \sqcap C_n)^{\mathcal{I}} & = & C_1^{\mathcal{I}} \cap ... \cap C_n^{\mathcal{I}}
\end{array}$$

In the following, the *satisfiability* condition for axioms and assertions of an $\mathcal{ALH}_f{}^-$-knowledge base Σ in an interpretation $\mathcal{I}$ are defined. A concept inclusion $C \sqsubseteq D$ (concept definition $C \equiv D$) is satisfied in $\mathcal{I}$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (resp. $C^{\mathcal{I}} = D^{\mathcal{I}}$) and a role inclusion $R \sqsubseteq S$ (role definition $R \equiv S$), if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ (resp. $R^{\mathcal{I}} = S^{\mathcal{I}}$). Similarly, assertions $C(a)$ and $R(a, b)$ are satisfied in $\mathcal{I}$, if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ resp. $(a, b)^{\mathcal{I}} \in R^{\mathcal{I}}$. If an interpretation $\mathcal{I}$ satisfies all axioms of $\mathcal{T}$ resp. $\mathcal{A}$ it is called a *model* of $\mathcal{T}$ resp. $\mathcal{A}$. If it satisfies both $\mathcal{T}$ and $\mathcal{A}$ it is called a model of Σ. Finally, if there is a model of Σ (i.e., a model for $\mathcal{T}$ and $\mathcal{A}$), then Σ is called satisfiable.

We are now able to define the entailment relation $\models$. A DL knowledge base Σ *logically entails* an assertion α (symbolically $\Sigma \models \alpha$) if α is satisfied in all models of Σ. For an Abox $\mathcal{A}$, we say $\Sigma \models \mathcal{A}$ if $\Sigma \models \alpha$ for all $\alpha \in \mathcal{A}$.

2.2 Preliminaries on Probabilistic Knowledge Representation

The basic notion of probabilistic knowledge representation formalisms is the so-called *random experiment.* A *random variable* X is a function assigning a value to the result of a random experiment. The random experiment itself is not represented, so random variables are functions without arguments, which return different values at different points of time. Possible values of a random variable comprise the so-called *domain* of the random variable. In the sequel, we will use *boolean* random variables, whose values can be either 1 or 0 (*true* or $false$, respectively).

Let $\vec{X} = \{X_1, ..., X_n\}$ be the ordered set of all random variables of a random experiment. An *event* (denoted $\vec{X} = \vec{x}$) is an assignment $X_1 = x_1, ..., X_n = x_n$ to all random variables. In case $n = 1$ we call the event simple, otherwise the event is called complex. A certain vector of values $\vec{x}$ is referred to as a *possible world.* A possible world can be associated with a *probability value* or *probability* for short. Hence, the notion of a possible world can be used as a synonym for an event, and depending on the context we use the former or the latter name. In case of an event with a boolean random variable X, we write x as an abbreviation for $X = true$ and $\neg x$ as an abbreviation for $X = false$.

Mappings of events to probabilities (or assignment of probabilities to events) are specified with so-called *probability assertions* of the following syntax: $P(\vec{X} = \vec{x}) = p$, where $\vec{X}$ is a vector of random variables, and p is a real value between 0 and 1 (it is assumed that the reader is familiar with Kolmogorov's axioms of probability). In the special case of a simple event (single random variable, $n = 1$) we write $P(X = x) = p$. The probability value p of an event is denoted as $P(\vec{X} = \vec{x})$ (or $P(X = x)$ in the simple case). In its raw form a set of probabilistic assertions is called a *probabilistic knowledge base* (with signature $\vec{X}$).

A mapping from the domain of a random variable X to probability values $[0, 1]$ is called a *distribution.* For distributions we use the notation $\mathbf{P}(X)$. Distributions can be defined for (ordered) sets of random variables as well. In this case we use $\mathbf{P}(X_1, \ldots, X_n)$ as a denotation for a mapping to the n-dimensional cross product of $[0, 1]$. For specifying a distribution, probability assertions for *all* domain values must be specified, and the values p must sum up to 1. In case all random variables of a random experiment are involved, we speak of a *(full) joint probability distribution* (JPD), otherwise the expression is said to denote a *marginal distribution* (projection of the n-dimensional space of probability values to a lower-dimensional space with m dimensions). The expression $\mathbf{P}(X_1, \ldots, X_m, X_{m+1} = x_{m+1}, \ldots, X_l = x_l)$ denotes an m-dimensional distribution with known values $x_{m+1}, \ldots, x_l$. In slight misuse of notation, we sometimes write $\vec{e}$ for these known values (e stands for evidence). The fragment $\vec{e}$ need not necessarily be written at the end in the parameter list of $\mathbf{P}$.

A *conditional probability* for a set of random variables $X_1, ..., X_m$ is denoted with $P(X_1 = x_1, ..., X_m = x_m \mid \vec{e})$ or, in distribution form, we write $\mathbf{P}(X_1, ..., X_m \mid \vec{e})$ (conditional probability distribution). This distribution can be also written as $\frac{\mathbf{P}(\vec{X},\vec{e})}{\mathbf{P}(\vec{e})}$.

For a probabilistic knowledge base, formal inference problems are defined. We restrict our attention to the two most convenient probabilistic inference problems: A *conditional probability query* is the computation of the joint distribution of a set of m random variables conditioned on $\vec{e}$ and is denoted with

$$P(x_1, ..., x_m \mid \vec{e}) = ?.$$

or, in distribution form:

$$\mathbf{P}(X_1, ..., X_m \mid \vec{e}) = ?.$$

The Maximum A Posteriori (MAP) inference returns the most-likely state of query atoms given the evidence. Based on the MAP inference, the "most probable world" given the evidence is determined as a set of events. The MAP inference problem given a distribution $\mathbf{P}$ specified for a set of random variables X is formalized as follows:

$$MAP_X(\vec{e}) := \vec{e} \cup argmax_{\vec{x}} P(\vec{x}|\vec{e}) \tag{1}$$

where $vars(\vec{x}) \cap vars(\vec{e}) = \emptyset$ and $vars(\vec{x}) \cup vars(\vec{e}) = X$ with $vars$ specified in the obvious way.

2.3 Markov Logic

The formalism of Markov logic [Domingos and Richardson, 2007] provides a means to combine the expressivity of first-order logic augmented with the formalism of Markov networks [Pearl, 1988]. The Markov logic formalism uses first-order logic to define "templates" for constructing Markov networks. The basic notion for this is a called a Markov logic network.

A Markov logic network $MLN = (\mathcal{F}_{MLN}, \mathcal{W}_{MLN})$ consists of an ordered multiset of first-order formulas $\mathcal{F_{MLN}} = \{F_1, ..., F_m\}$ and an ordered multiset of real number weights $\mathcal{W} = \{w_1, ..., w_m\}$. The association of a formula to its weight is by position in the ordered sets. For a formula $F \in \mathcal{F}_{MLN}$ with associated weight w we also write wF (weighted formula). Thus, a Markov logic network can also be defined as a set of weighted formulas. Both views can be used interchangeably. As a notational convenience, for ordered sets we nevertheless sometimes write $\vec{X}, \vec{Y}$ instead of $\vec{X} \cup \vec{Y}$.

In contrast to standard first-order logics such as predicate logic, relational structures not satisfying a formula F_i are not ruled out as models. If a relational structure does not satisfy a formula associated with a large weight it is just considered to be quite unlikely the "right" one.

Let $C = \{c_1, ..., c_m\}$ be the set of all constants mentioned in $\mathcal{F}_{MLN}$. A *grounding* of a formula $F_i \in \mathcal{F}_{MLN}$ is a substitution of all variables in the matrix of F_i with constants from C. From all groundings, the (finite) set of grounded atomic formulas (also referred to as *ground atoms*) can be obtained. Grounding corresponds to a domain closure assumption. The motivation is to get rid of the quantifiers and reduce inference problems to the propositional case.

Since a ground atom can either be true or false in an interpretation (or world), it can be considered as a boolean random variable X. Consequently, for each *MLN* with associated random variables $\vec{X}$, there is a set of possible worlds $\vec{x}$. In this view, sets of ground atoms are sometimes used to denote worlds. In this context, negated ground atoms correspond to *false* and non-negated ones to *true*. We denote worlds using a sequence of (possibly negated) atoms.

When a world $\vec{x}$ violates a weighted formula (does not satisfy the formula) the idea is to ensure that this world is less probable rather than impossible as in predicate logic. Note that weights do not directly correspond to probabilities (see [Domingos and Richardson, 2007] for details).

For each possible world of a Markov logic network $MLN = (\mathcal{F}_{MLN}, \mathcal{W}_{MLN})$ there is a probability for its occurrence. Probabilistic knowledge is required to obtain this value. As usual, probabilistic knowledge is specified using a probability distribution. In the formalism of Markov networks the full joint probability distribution of a Markov logic network *MLN* could be specified in symbolic form using a so-called log-linear form $P(\vec{X} = \vec{x}) = log\_lin(\vec{x})$ (see, e.g., [Domingos and Richardson, 2007]), with *log_lin* being defined as

$$log\_lin_{MLN}(\vec{x}) = \frac{1}{Z} exp\left(\sum_{i=1}^{|\mathcal{F}_{MLN}|} w_i n_i(\vec{x}) \right)$$

According to this definition, the probability of a possible world $\vec{x}$ is determined by the exponential of the sum of the number of true groundings (n_i) of formula $F_i \in \mathcal{F}_{MLN}$ in $\vec{x}$ multiplied with their corresponding weights $w_i \in \mathcal{W}_{MLN}$, and finally normalized with

$$Z = \sum_{\vec{x} \in \vec{X}} \exp\,(\sum_{i=1}^{|\mathcal{F}_{MLN}|} w_i n_i(\vec{x})), \tag{2}$$

the sum of the probabilities of all possible worlds. Thus, rather than specifying the full joint distribution directly in symbolic form as we have discussed before, in the Markov logic formalism, the probabilistic knowledge is specified implicitly by the weights associated with formulas. Determining these formulas and their weights in a practical context is all but obvious, such that machine learning techniques are usually employed for knowledge acquisition.

In the case of Markov logic, the definition of the *MAP* problem given in (1) can be rewritten as follows. The conditional probability term $P(\vec{x}|\vec{e})$ is replaced with with the Markovian formula:

$$MAP_{MLN}(\vec{e}) := \vec{e} \cup argmax_{\vec{x}} \frac{1}{Z_e} \exp\left(\sum_i w_i n_i\,(\vec{x}, \vec{e}) \right) \tag{3}$$

Thus, for describing the most-probable world, *MAP* returns a set of events, one for each random variable used in the Markov network derived from *MLN*. In the above equation, $\vec{x}$ denotes the hidden variables, and Z_e denotes the normalization constant which indicates that the normalization process is performed over possible worlds consistent with the evidence $\vec{e}$. In the next equation, Z_e is removed since it is constant and it does not affect the *argmax* operation. Similarly, in order to optimize the *MAP* computation the exp function is left out since it is a monotonic function and only its argument has to be maximized:

$$MAP_{MLN}(\vec{e}) := \vec{e} \cup argmax_{\vec{x}} \sum_i w_i n_i\,(\vec{x}, \vec{e}) \tag{4}$$

The above equation shows that the MAP problem in Markov logic formalism is reduced to a new problem which maximizes the sum of weights of satisfied clauses.

Since the MAP determination in Markov networks is an **NP**-hard problem [Domingos and Richardson, 2007], it is performed by exact and approximate solvers. The most commonly used approximate solver is MaxWalkSAT algorithm, a weighted variant of the WalkSAT local-search satisfiability solver. The MaxWalkSAT algorithm attempts to satisfy clauses with positive weights and keep clauses with negative weights unsatisfied.

It has to be mentioned that there might be several worlds with the same maximal probability. But at this step, only one of them is chosen non-deterministically.

2.4 Combining Markov Logic and Description Logic

Since $\mathcal{ALH}_f{}^-$ is a fragment of first-order logic, its extension to the Markovian style of formalisms is specified in a similar way as for predicate logic in the section before. The formulas in Markov logic correspond to Tbox axioms and Abox assertions. Weights in Markov description logics are associated with axioms and assertions.

Groundings of Tbox axioms are defined analogously to the previous case.[1] Abox assertions do not contain variables and are already grounded. Note that since an $\mathcal{ALH}_f{}^-$ Abox represents a relational structure of domain objects, it can be directly seen as a possible world itself if assertions not contained in the Abox are assumed to be false.

For appropriately representing domain knowledge in CASAM, weights are possibly used only for a subset of the axioms of the domain ontology. The remaining axioms can be assumed to be *strict*, i.e., assumed to be true in any case. A consequence of specifying strict axioms is that lots of possible worlds $\vec{x}$ can be ruled out (i.e., will have probability 0 by definition).

A *Markov DL knowledge base* Σ_M is a tuple $(\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is comprised of a set $\mathcal{T}_s$ of strict axioms and a set $\mathcal{T}_w$ of weighted axioms and $\mathcal{A}$ is comprised of a set $\mathcal{A}_s$ of strict assertions and a

[1]For this purpose, the variable-free syntax of axioms can be first translated to predicate logic.

set $\mathcal{A}_w$ of weighted assertions. Referring to axioms, a proposal for CASAM is to consider strictness for the domain ontology patterns (I)–(IV):

(I)	subsumption	$A_1 \sqsubseteq A_2,\ R_1 \sqsubseteq R_2$
(II)	disjointness	$A_1 \sqsubseteq \neg A_2$
(III)	domain and range restrictions	$\exists R.\top \sqsubseteq A,\ \top \sqsubseteq \forall R.A$
(IV)	functional roles	$\top \sqsubseteq (\leq 1\, R)$

The main justification treating axioms as strict is that the subsumption axioms, disjointness axioms, domain and range restrictions as well as functional role axioms (in combination with UNA) are intended to be true in any case such that there is no need to assign large weights to them.

The advantage of this probabilistic approach is that initial ontology engineering is done as usual with standard reasoning support and with the possibility to add weighted axioms and weighted assertions on top of the strict fundament. Since lots of possible worlds do not have to be considered because their probability is known to be 0, probabilistic reasoning will be significantly faster.

3 Probabilistic Interpretation Engine

In this chapter, the conceptual architecture of RMI is presented. The architecture has been slightly extended in comparison to the architecture given in Deliverable D3.2. Some definitions are also given which are required for explaining the subjects mentioned in this chapter including the abduction procedure and the interpretation procedure. The abduction procedure is defined by the abduction algorithm CAE. Additionally, the interpretation procedure is described by presenting a probabilistic interpretation algorithm *Interpret*. The other important point in comparison to Deliverable D3.2 is that we define a termination condition for the abduction process. The functionality of the RMI agent is also described by an algorithm. Finally, the optimization techniques for the probabilistic abduction engine are introduced.

3.1 RMI Conceptual Architecture

Figure 1 depicts the basic components at the conceptual level with input and output data (Aboxes in both cases), intermediate results (Aboxes) as well as the background knowledge (a Tbox and a set of rules) used by different modules.[2]

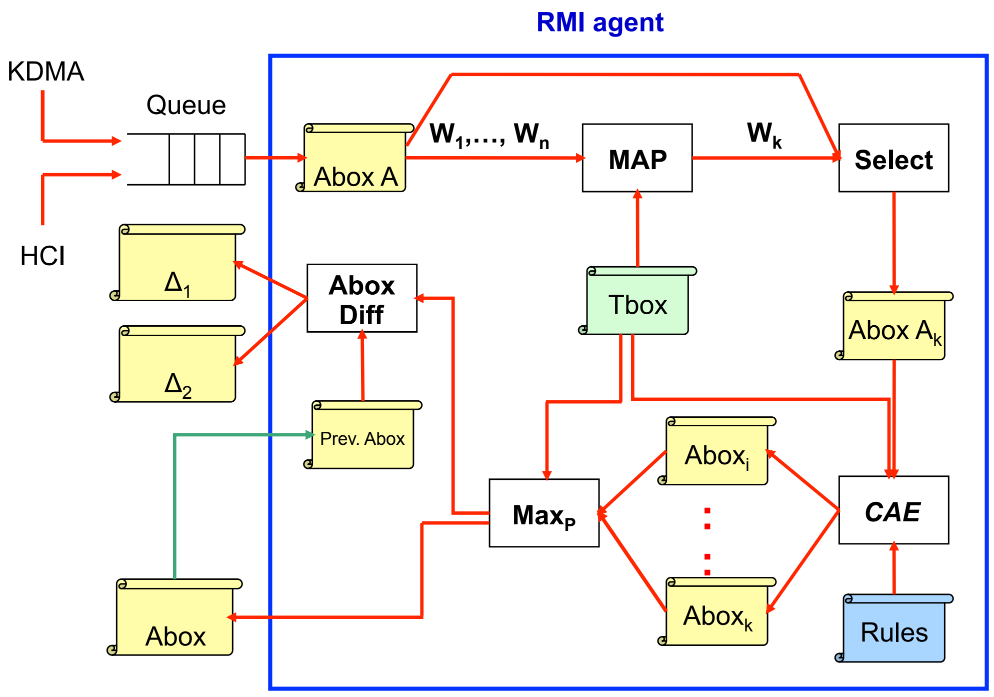

Figure 1: Conceptual view of the reasoning-based media interpretation engine.

The above figure is described as follows: the KDMA and HCI components send their percept results incrementally to a queue Q. RMI extracts these results from the queue using a focus strategy (e.g., process a video shot by shot and start over with the first shot if the last one is focused on) and transforms them into an Abox $\mathcal{A}$. Since it is possible that the Abox $\mathcal{A}$ is inconsistent, the MAP process is applied to select a consistent subset A_k The MAP process determines the most probable world based on the Markov logic formalism as explained in Chapter 2. W_k is composed of positive

[2]The background knowledge could also contain Abox assertions to represent background knowledge about individuals. We neglect this here, however, in order to simplify the presentation.

and negative assertions. The functionality of the next unit called *Select* is to choose only the positive assertions of W_k since only these assertions require explanations. The generated Abox $\mathcal{A}_k$ is sent to the unit CAE which is the core component of the above interpretation engine. By applying a set of rules and the Tbox $\mathcal{T}$, a set of interpretation Aboxes are generated. The functionality of the CAE unit is explained in more detail in Section 3.2.5. Afterwards, one of the generated Aboxes is selected. The functionality of the MAX_p unit is to select the Abox whose most-probable world has the highest probability. Given the interpretation Abox that gets the best score using this scheme and the one previously obtained, an Abox difference operation is used to compute the Abox differences Δ_1 and Δ_2 for additions and omissions, respectively. RMI returns the currently best Abox for the focus as well as Δ_1 and Δ_2 (and stores the currently best Abox for the observations in focus).

3.2 Implementation of the RMI Agent

3.2.1 Sequences, Variable Substitutions and Transformations

A *variable* is a name of the form *String* where *String* is a string of characters from {A...Z}. In the following definitions, we denote places where variables can appear with uppercase letters.

Let V be a set of variables, and let $\underline{X}, \underline{Y_1}, \ldots, \underline{Y_n}$ be sequences $\langle \ldots \rangle$ of variables from V. $\underline{z}$ denotes a sequence of individuals. We consider sequences of length 1 or 2 only, if not indicated otherwise, and assume that $(\langle X \rangle)$ is to be read as (X) and $(\langle X, Y \rangle)$ is to be read as (X, Y) etc. Furthermore, we assume that sequences are automatically flattened. A function *as_set* turns a sequence into a set in the obvious way.

A *variable substitution* $\sigma = [X \leftarrow i, Y \leftarrow j, \ldots]$ is a mapping from variables to individuals. The application of a variable substitution σ to a sequence of variables $\langle X \rangle$ or $\langle X, Y \rangle$ is defined as $\langle \sigma(X) \rangle$ or $\langle \sigma(X), \sigma(Y) \rangle$, respectively, with $\sigma(X) = i$ and $\sigma(Y) = j$. In this case, a sequence of individuals is defined. If a substitution is applied to a variable X for which there exists no mapping $X \leftarrow k$ in σ then the result is undefined. A variable for which all required mappings are defined is called *admissible* (w.r.t. the context).

3.2.2 Grounded Conjunctive Queries

Let $\underline{X}, \underline{Y_1}, \ldots, \underline{Y_n}$ be sequences of variables, and let $Q_1, \ldots, Q_n$ denote concept or role names.

A query is defined by the following syntax.

$$\{(\underline{X}) \mid Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})\}$$

The sequence $\underline{X}$ may be of arbitrary length but all variables mentioned in $\underline{X}$ must also appear in at least one of the $\underline{Y_1}, \cdots, \underline{Y_n}$: $as\_set(\underline{X}) \subseteq as\_set(\underline{Y_1}) \cup \cdots \cup as\_set(\underline{Y_n})$.

Informally speaking, $Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})$ defines a conjunction of so-called *query atoms* $Q_i(\underline{Y_i})$. The list of variables to the left of the sign | is called the *head* and the atoms to the right of are called the query *body*. The variables in the head are called distinguished variables. They define the query result. The variables that appear only in the body are called non-distinguished variables and are existentially quantified.

Answering a query with respect to a knowledge base Σ means finding admissible variable substitutions σ such that $\Sigma \models \{\sigma(Q_1(\underline{Y_1})), \ldots, \sigma(Q_n(\underline{Y_n}))\}$. We say that a variable substitution $\sigma = [X \leftarrow i, Y \leftarrow j, \ldots]$ introduces *bindings* $i, j, \ldots$ for variables $X, Y, \ldots$. Given all possible variable substitutions σ, the *result* of a query is defined as $\{(\sigma(\underline{X}))\}$. Note that the variable substitution σ is applied before checking whether $\Sigma \models \{Q_1(\sigma(\underline{Y_1})), \ldots, Q_n(\sigma(\underline{Y_n}))\}$, i.e., the query is *grounded* first.

For a query $\{(?y) \mid Person(?x), hasParticipant(?y, ?x)\}$ and the Abox $\Gamma_1 = \{HighJump(ind_1), Person(ind_2), hasParticipant(ind_1, ind_2)\}$, the substitution $[?x \leftarrow ind_2, ?y \leftarrow ind_1]$ allows for answering the query, and defines bindings for $?y$ and $?x$.

A *boolean* query is a query with $\underline{X}$ being of length zero. If for a boolean query there exists a variable substitution σ such that $\Sigma \models \{\sigma(Q_1(\underline{Y_1})), \ldots, \sigma(Q_n(\underline{Y_n}))\}$ holds, we say that the query is answered with *true*, otherwise the answer is *false*.

Later on, we will have to convert query atoms into Abox assertions. This is done with the function *transform*. The function *transform* applied to a set of query atoms $\{\gamma_1, \ldots \gamma_n\}$ is defined

as $\{transform(\gamma_1, \sigma), \ldots, transform(\gamma_n, \sigma)\}$ where
$transform(P(\underline{X}), \sigma) := P(\sigma(\underline{X}))$.

3.2.3 Rules

A rule r has the following form $P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})$ where P, $Q_1, \ldots, Q_n$ denote concept or role names with the additional restriction (safety condition) that $as\_set(\underline{X}) \subseteq as\_set(\underline{Y_1}) \cup \cdots \cup as\_set(\underline{Y_n})$.
Rules are used to derive new Abox assertions, and we say that a rule r is *applied* to an Abox $\mathcal{A}$. The function call $apply(\Sigma, P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n}), \mathcal{A})$ returns a set of Abox assertions $\{\sigma(P(\underline{X}))\}$ if there exists an admissible variable substitution σ such that the answer to the query

$$\{() \mid Q_1(\sigma(\underline{Y_1})), \ldots, Q_n(\sigma(\underline{Y_n}))\}$$

is *true* with respect to $\Sigma \cup \mathcal{A}$.[3] If no such σ can be found, the result of the call to $apply(\Sigma, r, \mathcal{A})$ is the empty set. The application of a set of rules $\mathcal{R} = \{r_1, \ldots r_n\}$ to an Abox is defined as follows.

$$apply(\Sigma, \mathcal{R}, \mathcal{A}) = \bigcup_{r \in \mathcal{R}} apply(\Sigma, r, \mathcal{A})$$

The result of $forward\_chain(\Sigma, \mathcal{R}, \mathcal{A})$ is defined to be $\emptyset$ if $apply(\Sigma, \mathcal{R}, \mathcal{A}) \cup \mathcal{A} = \mathcal{A}$ holds. Otherwise the result of *forward_chain* is determined by the recursive call
$apply(\Sigma, \mathcal{R}, \mathcal{A}) \cup forward\_chain(\Sigma, \mathcal{R}, \mathcal{A} \cup apply(\Sigma, \mathcal{R}, \mathcal{A}))$.

For some set of rules $\mathcal{R}$ we extend the entailment relation by specifying that $(\mathcal{T}, \mathcal{A}) \models_{\mathcal{R}} \mathcal{A}_0$ iff $(\mathcal{T}, \mathcal{A} \cup forward\_chain((\mathcal{T}, \emptyset), \mathcal{R}, \mathcal{A})) \models \mathcal{A}_0$.[4]

3.2.4 Computing Explanations via Abduction

In general, abduction is formalized as $\Sigma \cup \Delta \models_{\mathcal{R}} \Gamma$ where background knowledge (Σ), rules ($\mathcal{R}$), and observations (Γ) are given, and explanations (Δ) are to be computed. In terms of DLs, Δ and Γ are Aboxes and Σ is a pair of Tbox and Abox.

Abox abduction is implemented as a non-standard retrieval inference service in DLs. In contrast to standard retrieval inference services where answers are found by exploiting the ontology, Abox abduction has the task of acquiring what should be added to the knowledge base in order to answer a query. Therefore, the result of Abox abduction is a set of hypothesized Abox assertions. To achieve this, the space of abducibles has to be previously defined and we do this in terms of rules.

We assume that a set of rules $\mathcal{R}$ as defined above (see Section 3.2.3) are specified, and define a non-deterministic function *compute_explanation* as follows.

- $compute\_explanation(\Sigma, \mathcal{R}, \mathcal{A}, P(\underline{z})) = transform(\Phi, \sigma)$ if there exists a rule $r = P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n}) \in \mathcal{R}$ that is applied to an Abox $\mathcal{A}$ such that a minimal set of query atoms Φ and an admissible variable substitution σ with $\sigma(\underline{X}) = \underline{z}$ can be found, and the query $Q := \{() \mid expand(P(\underline{z}), r, \mathcal{R}, \sigma) \setminus \Phi\}$ is answered with *true*.

- If no such rule r exists in $\mathcal{R}$ it holds that $compute\_explanation(\Sigma, \mathcal{R}, \mathcal{A}, P(\underline{z})) = \emptyset$.

The goal of the function *compute_explanation* is to determine what must be added (Φ) such that an entailment $\Sigma \cup \mathcal{A} \cup \Phi \models_{\mathcal{R}} P(\underline{Z})$ holds. Hence, for *compute_explanation*, abductive reasoning is used. The set of query atoms Φ defines what must be hypothesized in order to answer the query Q with *true* such that $\Phi \subseteq expand(P(\underline{X}), r, \mathcal{R}, \sigma)$ holds. The definition of *compute_explanation* is non-deterministic due to several possible choices for Φ.

[3]We slightly misuse notation in assuming $(\mathcal{T}, \mathcal{A}) \cup \Delta = (\mathcal{T}, \mathcal{A} \cup \Delta)$. If $\Sigma \cup \mathcal{A}$ is inconsistent the result is well-defined but useless. It will not be used afterwards.

[4]We could also give a semantic definition of entailment w.r.t. a set of rules without using *forward_chain*. However, in this deliverable we do not attempt to prove that the abduction algorithm is correct. Thus, only proof-theoretic definition is given.

The function application $expand(P(\underline{Z}), P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n}), \mathcal{R})$ is also defined in a non-deterministic way as

$$expand'(Q_1(\sigma'(\underline{Y_1})), \mathcal{R}, \sigma) \cup \cdots \cup expand'(Q_n(\sigma'(\underline{Y_n})), \mathcal{R}, \sigma)$$

with $expand'(P(\underline{Z}), \mathcal{R}, \sigma)$ being $expand(P(\sigma'(\underline{z})), r, \mathcal{R}, \sigma')$ if there exist a rule $r = P(\underline{X}) \leftarrow \ldots \in \mathcal{R}$ and $\langle P(\underline{X}) \rangle$ otherwise. The variable substitution σ' is an extension of σ such that:

$$\sigma' = [X_1 \leftarrow z_1, X_2 \leftarrow z_2, \ldots] \tag{5}$$

The above equation shows the mapping of the free variables if it is not already defined. This means the free variables in the body of each rule are mapped to individuals with unique IDs.

We say the set of rules is backward-chained, and since there might be multiple rules in $\mathcal{R}$, backward-chaining is non-deterministic as well. Thus, multiple explanations are generated.[5]

3.2.5 The Abduction Procedure

In the following, we devise an abstract computational engine for "explaining" Abox assertions in terms of a given set of rules. Explanation of Abox assertions w.r.t. a set of rules is meant in the sense that using the rules some high-level explanations are constructed such that the Abox assertions are entailed. The explanation of an Abox is again an Abox. For instance, the output Abox represents results of the RMI content interpretation process. The presentation in slightly extended compared to the one in [Castano et al., 2008]. Let the agenda $\mathfrak{A}$ be a set of Aboxes Γ and let Γ be an Abox of observations whose assertions are to be explained. The goal of the explanation process is to use a set of rules $\mathcal{R}$ to derive "explanations" for elements in Γ. The explanation algorithm implemented in the CASAM abduction engine works on a set of Aboxes $\mathfrak{I}$.
The complete explanation process is implemented by the CAE function:

```
**Function** CAE($\Omega$, $\Xi$, $\Sigma$, $\mathcal{R}$, $S$, $\mathfrak{A}$):
**Input:** a strategy function $\Omega$, a termination function $\Xi$, a background knowledge $\Sigma$, a set of
rules $\mathcal{R}$, a scoring function $S$, and an agenda $\mathfrak{A}$
**Output:** a set of interpretation Aboxes $\mathfrak{I}'$
$\mathfrak{I}' := \{assign\_level(l, \mathfrak{A})\}$;
medskip **repeat**
    $\mathfrak{I} := \mathfrak{I}'$;
    $(\mathcal{A}, \alpha) := \Omega(\mathfrak{I})$  // Select $\mathcal{A} \in \mathfrak{I}$, $\alpha \in \mathcal{A}$ according to the strategy function $\Omega$;
    $l = l + 1$;
    $\mathfrak{I}' := (\mathfrak{A} \setminus \{\mathcal{A}\}) \cup assign\_level(l, explanation\_step(\Sigma, \mathcal{R}, S, \mathcal{A}, \alpha))$;
**until** $\Xi(\mathfrak{I})$ *or no* $\mathcal{A}$ *and* $\alpha$ *can be selected such that* $\mathfrak{I}' \neq \mathfrak{I}$ ;
**return** $\mathfrak{I}'$
```

where $assign\_level(l, \mathfrak{A})$ is defined by a lambda calculus term as follows:

$$assign\_level(l, \mathfrak{A}) = map(\lambda(\mathcal{A}) \bullet assign\_level(l, \mathcal{A}), \mathfrak{A}) \tag{6}$$

$assign\_level(l, \mathfrak{A})$ takes as input a superscript l and an agenda $\mathfrak{A}$.

In the following, $assign\_level(l, \mathcal{A})$ is defined which superscripts each assertion α of the Abox $\mathcal{A}$ with l if the assertion α does not already have a superscript:

$$assign\_level(l, \mathcal{A}) = \left\{\alpha^l \mid \alpha \in \mathcal{A}, \alpha \neq \beta^i, i \in \mathbb{N}\right\} \tag{7}$$

Note that l is a global variable, its starting value is zero and it is incremented in the CAE function. The map[6] function is defined as follows:

$$map(f, X) = \bigcup_{x \in X} \{f(x)\} \tag{8}$$

[5] In the expansion process, variables have to be renamed. We neglect these issues here.

[6] Please note that in this report, the expression *map* is used in two different contexts. The first one MAP denotes the Maximum A Posteriori approach which is a sampling method whereas the second one *map* is a function used in the $assign\_level(l, \mathfrak{A})$ function.

It takes as parameters a function f and a set X and returns a set consisting of the values of f applied to every element x of X.
CAE function applies the strategy function Ω in order to decide which assertion to explain, uses a termination function Ξ in order to check whether to terminate due to resource constraints and a scoring function S to evaluate an explanation.

The function Ω for the explanation strategy and Ξ for the termination condition are used as an oracle and must be defined in an application-specific way. The function *explanation_step* is defined as follows.

$explanation\_step(\Sigma, \mathcal{R}, S, \mathcal{A}, \alpha)$:

$$\bigcup_{\Delta \in compute\_all\_explanations(\Sigma,\mathcal{R},S,\mathcal{A},\alpha)} consistent\_completed\_explanations(\Sigma, \mathcal{R}, \mathcal{A}, \Delta).$$

We need two additional auxiliary functions.

$consistent\_completed\_explanations(\Sigma, \mathcal{R}, \mathcal{A}, \Delta)$:

$$\{\Delta' \mid \Delta' = \Delta \cup \mathcal{A} \cup forward\_chain(\Sigma, \mathcal{R}, \Delta \cup \mathcal{A}), consistent_{\Sigma}(\Delta')\}$$

$compute\_all\_explanations(\Sigma, \mathcal{R}, S, \mathcal{A}, \alpha)$:

$$maximize(\Sigma, \mathcal{R}, \mathcal{A}, \{\Delta \mid \Delta = compute\_explanation(\Sigma, \mathcal{R}, \alpha), consistent_{\Sigma\cup\mathcal{A}}(\Delta)\}, S).$$

The function $maximize(\Sigma, \mathcal{R}, \mathcal{A}, \Delta s, S)$ selects those explanations $\Delta \in \Delta$s for which the score $S(\Sigma, \mathcal{R}, \mathcal{A}, \Delta)$ is maximal, i.e., there exists no other $\Delta' \in \Delta s$ such that $S(\Sigma, \mathcal{R}, \mathcal{A}, \Delta') > S(\Sigma, \mathcal{R}, \mathcal{A}, \Delta)$. The function $consistent_{(\mathcal{T},\mathcal{A})}(\mathcal{A}')$ determines if the Abox $\mathcal{A} \cup \mathcal{A}'$ has a model which is also a model of the Tbox $\mathcal{T}$.

Note the call to the nondeterministic function *compute_explanation*. It may return different values, all of which are collected.

In the next Section we explain how probabilistic knowledge is used to (i) formalize the effect of the “explanation”, and (ii) formalize the scoring function S used in the CAE algorithm explained above. In addition, it is shown how the termination condition (represented with the parameter Ξ in the above procedure) can be defined based on the probabilistic conditions.

3.2.6 The Interpretation Procedure

The interpretation procedure is completely discussed in this section by explaining the interpretation problem and presenting a solution to this problem. The solution is presented by a probabilistic interpretation algorithm which calls the CAE function described in the previous section. In the given algorithm, a termination function, and a scoring function are defined. The termination function determines if the interpretation process can be stopped since at some point during the interpretation process it makes no sense to continue the process. The reason for stopping the interpretation process is that no significant changes can be seen in the results. The defined scoring function in this section assigns scores to interpretation Aboxes.

Problem The objective of the RMI component is the generation of interpretations for the observation resluts determined by the KDMA and HCI components. An interpretation is an Abox which contains high level concept assertions. Since in the artificial intelligence, the agents are used for solving the problems, in the following the same problem is formalized in the perspective of an agent:
Consider an intelligent agent and some percepts in an environment where the percepts are the analysis results of KDMA and HCI. The objective of this agent is finding explanations for the existence of percepts. The question is how the interpretation Aboxes are determined and how long the interpretation process must be performed by the agent. The functionality of this agent is presented in the *RMI_Agent* algorithm in Section 3.2.7.

Solution In the following, an application for a probabilistic interpretation algorithm is presented which gives a solution to the mentioned problem. This solution illustrates a new perspective to the interpretation process and the reason why it is performed. Assume that the RMI component receives a weighted Abox $\mathcal{A}$ from KDMA and HCI which contains observations. In the following, the applied operation $P(\mathcal{A}, \mathcal{A}', \mathcal{R}, \mathcal{WR}, \mathcal{T})$ in the algorithm is explained:

The $P(\mathcal{A}, \mathcal{A}', \mathcal{R}, \mathcal{WR}, \mathcal{T})$ function determines the probability of the Abox $\mathcal{A}$ with respect to the Abox $\mathcal{A}'$, a set of rules $\mathcal{R}$, a set of weighted rules $\mathcal{WR}$, and the Tbox $\mathcal{T}$ where $\mathcal{A} \subseteq \mathcal{A}'$. Note that $\mathcal{R}$ is a set of forward and backward chaining rules. The probability determination is performed based on the Markov logic formalism as follows:

$$P(\mathcal{A}, \mathcal{A}', \mathcal{R}, \mathcal{WR}, \mathcal{T}) = P_{MLN(\mathcal{A}, \mathcal{A}', \mathcal{R}, \mathcal{WR}, \mathcal{T})}(\vec{Q}(\mathcal{A}) \mid \vec{e}(\mathcal{A}')) \tag{9}$$

$\vec{Q}(\mathcal{A})$ denotes the fulljoint probability of all assertions which appear in the Abox $\mathcal{A}$. Assume Abox $\mathcal{A}$ contains n assertions $\alpha_1, \ldots, \alpha_n$. Consequently, the query of the Abox $\mathcal{A}$ is defined as follows:

$$\vec{Q}(\mathcal{A}) = \{\langle \alpha_1 \wedge \ldots \wedge \alpha_n \rangle \mid \alpha_i \in \mathcal{A}, i \in \mathbb{N}\} \tag{10}$$

and assume that the Abox $\mathcal{A}'$ contains m assertions $\alpha_1, \ldots, \alpha_m$, so that $m \geq n$. In the following, the evidence vector $\vec{e}(\mathcal{A}')$ is defined:

$$\vec{e}(\mathcal{A}') = \{\langle \alpha_1, \ldots, \alpha_m \rangle \mid \alpha_i \in \mathcal{A}', i \in \mathbb{N}\} \tag{11}$$

In order to answer the query $P(\mathcal{A}, \mathcal{A}', \mathcal{R}, \mathcal{WR}, \mathcal{T})$, the Markov logic network MLN based on the Aboxes $\mathcal{A}$ and $\mathcal{A}'$, the rules $\mathcal{R}$, the weighted rules $\mathcal{WR}$ and the Tbox $\mathcal{T}$ should be built which is a time consuming process. Note that the above function is called not only once but several times. In the following, the interpretation algorithm *Interpret* is presented:

```
**Function** Interpret($\mathfrak{A}$, $CurrentI$, $\Gamma$, $\mathcal{T}$, $\mathcal{FR}$, $\mathcal{BR}$, $\mathcal{WR}$, $\epsilon$)
**Input:** an agenda $\mathfrak{A}$, a current interpretation Abox $CurrentI$, an Abox of observations $\Gamma$, a Tbox $\mathcal{T}$, a set of forward chaining rules $\mathcal{FR}$, a set of backward chaining rules $\mathcal{BR}$, a set of weighted rules $\mathcal{WR}$, and the desired precision of the results $\epsilon$
**Output:** an agenda $\mathfrak{A}'$, a new interpretation Abox $NewI$, and Abox differences for additions $\Delta_1$ and omissions $\Delta_2$
$i := 0$ ;
$p_0 := P(\Gamma, \Gamma, \mathcal{R}, \mathcal{WR}, \mathcal{T})$ ;
$\Xi := \lambda(\mathfrak{A}) \bullet \{i := i + 1; p_i := \max_{\mathcal{A} \in \mathfrak{A}} P(\Gamma, \mathcal{A} \cup \mathcal{A}_0, \mathcal{R}, \mathcal{WR}, \mathcal{T}); \textbf{return} \mid p_i - p_{i-1} \mid < \frac{\epsilon}{i}\}$;
$\Sigma := (\mathcal{T}, \emptyset)$;
$\mathcal{R} := \mathcal{FR} \cup \mathcal{BR}$;
$S := \lambda((\mathcal{T}, \mathcal{A}_0)), \mathcal{R}, \mathcal{A}, \Delta) \bullet P(\Gamma, \mathcal{A} \cup \mathcal{A}_0 \cup \Delta, \mathcal{R}, \mathcal{WR}, \mathcal{T})$;
$\mathfrak{A}' := CAE(\Omega, \Xi, \Sigma, \mathcal{R}, S, \mathfrak{A})$;
$NewI = argmax_{\mathcal{A} \in \mathfrak{A}'}(P(\Gamma, \mathcal{A}, \mathcal{R}, \mathcal{WR}, \mathcal{T}))$;
$\Delta_1 = AboxDiff(NewI, CurrentI)$; // additions
$\Delta_2 = AboxDiff(CurrentI, NewI)$; // omissions
**return** $(\mathfrak{A}', NewI, \Delta_1, \Delta_2)$;
```

In the above algorithm, the termination function Ξ and the scoring function S are defined by lambda calculus terms. The termination condition Ξ of the algorithm is that no significant changes can be seen in the successive probabilities p_i and p_{i-1} (scores) of the two successive generated interpretation Aboxes in two successive levels $i - 1$ and i. In this case, the current interpretation Abox *CurrentI* is preferred to the new interpretation Abox *NewI*. In the next step, the CAE function is called which returns agenda $\mathfrak{A}'$. Afterwards, the interpretation Abox *NewI* with the maximum score among the Aboxes $\mathcal{A}$ of $\mathfrak{A}'$ is selected. Additionally, the Abox differences Δ_1 and Δ_2 respectively for additions and omissions among the interpretation Aboxes *CurrentI* and *NewI* are calculated. In Chapter 5, it is completely explained how the Abox difference process among two Aboxes is performed. In the following, the strategy condition Ω is defined which is one of the parameters of CAE function:

```
**Function** $\Omega(\mathfrak{I})$
**Input:** a set of interpretation Aboxes $\mathfrak{I}$
**Output:** an Abox $\mathcal{A}$ and a fiat assertion $\alpha$
$\mathfrak{A} := \left\{\mathcal{A} \in \mathfrak{I} \mid \neg\exists \mathcal{A}' \in \mathfrak{I}, \mathcal{A}' \neq \mathcal{A} : \exists {\alpha'}^{l'} \in \mathcal{A}' : \forall \alpha^{l} \in \mathcal{A} : l' < l\right\}$;
$\mathcal{A} := random\_select(\mathfrak{A})$;
$min\_\alpha_s = \left\{\alpha^{l} \in \mathcal{A} \mid \neg\exists {\alpha'}^{l'} \in \mathcal{A}', {\alpha'}^{l'} \neq \alpha^{l}, l' < l\right\}$;
**return** $(\mathcal{A}, random\_select(\{min\_\alpha_s\}))$;
```

In the above strategy function Ω, the agenda $\mathfrak{A}$ is a set of Aboxes $\mathcal{A}$ such that the assigned superscripts to their assertions are minimum. In the next step, an Abox $\mathcal{A}$ from $\mathfrak{A}$ is randomly selected. Afterwards, the min_α_s set is determined which contains the assertions α from $\mathcal{A}$ whose superscripts are minimum. These are the assertions which require explanations. The strategy function returns as output an Abox $\mathcal{A}$ and an assertion α which requires explanation.

3.2.7 The RMI Agent

In the following, the *RMI_Agent* function is presented which calls the *Interpret* function:

```
**Function** RMI_Agent($Q$, $Partners$, $Die$, $(\mathcal{T}, \mathcal{A}_0), \mathcal{FR}, \mathcal{BR}, \mathcal{WR}, \epsilon$)
**Input:** a queue of percept results $Q$, a set of partners $Partners$, a function $Die$ for the
terminatation process, a background knowledge set $(\mathcal{T}, \mathcal{A}_0)$, a set of forward chaining rules
$\mathcal{FR}$, a set of backward chaining rules $\mathcal{BR}$, a set of weighted rules $\mathcal{WR}$, and the desired
precision of the results $\epsilon$
**Output:** –
$CurrentI = \emptyset$;
$\mathfrak{A}'' = \{\emptyset\}$;
**repeat**
    $\Gamma := extractObservations(Q)$;
    $W := MAP(\Gamma, \mathcal{WR}, \mathcal{T})$ ;
    $\Gamma' := Select(W, \Gamma)$;
    $\mathfrak{A}' := filter(\lambda(\mathcal{A}) \bullet consistent_{\Sigma}(\mathcal{A}), map(\lambda(\mathcal{A}) \bullet \mathcal{A} \cup forward\_chain(\Sigma, \mathcal{FR}, \mathcal{A} \cup \mathcal{A}_0),$
        $\{select(MAP(\Gamma', \mathcal{WR}, \mathcal{T}, \mathcal{A}_0 \cup \mathcal{A}), \Gamma' \cup \mathcal{A}_0 \cup \mathcal{A}))) \mid \mathcal{A} \in \mathfrak{A}''\}$;
    $(\mathfrak{A}'', NewI, \Delta_1, \Delta_2) := Interpret(\mathfrak{A}', CurrentI, \Gamma', \mathcal{T}, \mathcal{FR}, \mathcal{BR}, \mathcal{WR} \cup \Gamma, \epsilon)$;
    $CurrentI := NewI$;
    $Communicate(\Delta_1, \Delta_2, Partners)$;
**until** $Die()$ ;
```

where the *filter* function is defined as follows:

$$filter(f, X) = \bigcup_{x \in X} \begin{cases} \{x\} & \text{if } f(x) = true \\ \emptyset & \text{else} \end{cases} \tag{12}$$

The *filter* function takes as parameters a function f and a set X and returns a set consisting of the values of f applied to every element x of X.

In the *RMI_Agent* function, the current interpretation $CurrentI$ and the agenda $\mathfrak{A}''$ are initialized to empty set. Since the agent performance is an incremental process, it is defined by a *repeat* – *until* loop. The percept results Γ are sent by KDMA and HCI to the queue Q. In order to take the observations Γ from the queue Q, the *RMI_Agent* calls the *extractObservations* function.
The $MAP(\Gamma, \mathcal{WR}, \mathcal{T}, \mathcal{A})$ function determines the most probable world of observations Γ with respect to a set of weighted rules $\mathcal{WR}$ and the Tbox $\mathcal{T}$. This function performs actually the mentioned MAP process in Chapter 2. It returns a vector W which consists of a set of zeros and ones assigned to the ground atoms of the considered world. The assertions with assigned zeros and ones are called respectively, negative and positive assertions.
The $Select(W, \Gamma)$ function selects the positive assertions from the bit vector W in the input Abox

Γ. The selected positive assertions which require explanations are also known as fiat assertions. This operation returns as output an Abox Γ' which has the following characteristic: $\Gamma' \subseteq \Gamma$.
In the next step, a set of forward chaining rules $\mathcal{FR}$ is applied to all the Aboxes of $\mathfrak{A}''$. The generated assertions in this process are added to the to the Abox $\mathcal{A}$. In the next step, only the consistent Aboxes are selected and the other inconsistent Aboxes are not considered for the next steps.

In the next step, the *Interpret* function is called to determine the new agenda $\mathfrak{A}''$, the new interpretation $NewI$ and the Abox differences Δ_1 and Δ_2 for additions and omissions among $CurrentI$ and $NewI$. Afterwards, the $CurrentI$ is set to the $NewI$ and the RMI_Agent function communicates the Abox differences Δ_1 and Δ_2 to the partners. Additionally, the Tbox $\mathcal{T}$, the set of forward chaining rules $\mathcal{FR}$, the set of backward chaining rules $\mathcal{BR}$, and the set of weighted rules $\mathcal{WR}$ can be learnt by the *Learn* function. The termination condition of the RMI_Agent function is that the $Die()$ function is true.

Note that the RMI_Agent waits at the function call $\mathit{extractObservations}(Q)$ if $Q = \emptyset$.

After presenting the above algorithms, the mentioned unanswered questions can be discussed. A reason for performing the interpretation process and explaining the fiat assertions is that the probability of $P(\mathcal{A}, \mathcal{A}', \mathcal{R}, \mathcal{WR}, \mathcal{T})$ will increase through the interpretation process. In other words, by explaining the observations the agent's belief to the percepts will increase. This shows a new perspective for performing the interpretation process.

The answer to the question whether there is any measure for stopping the interpretation process, is indeed positive. This is expressed by $\mid p_i - p_{i-1} \mid < \frac{\epsilon}{i}$ which is the termination condition Ξ of the algorithm. The reason for selecting $\frac{\epsilon}{i}$ and not ϵ as the upper limit for the termination condition is to terminate the oscillation behaviour of the results. In other words, the precision interval is tightened step by step during the interpretation process.

3.3 Additional Techniques

In the description of the concepual architecture of RMI we have discussed that the RMI agent focuses on observations with respect to single video shots (see Chapter 3). In the detailed description however, we have not completely formalized this. In addition, even the following extension can be realized.

Since the topic of the video shots does not differ completely from each other, we have to consider the high level concept assertions of the previous video shots for the next video shots. Let us assume that a video which shows an interview is analysed. If a concept assertion $interview(ind_1)$ is hypothesized for a video shot, it probably makes no sense to generate new individuals of the type *interview* for the next video shots. This means the individual ind_1 will be reused for the next video shots. More formally, assume the video shots S_1 and S_2 from the same video and consider Γ_{H1} as the high level concept assertions of S_1. If Γ_2 denotes the assertions related to the video shot S_2, then the assertions which are considered for the interpretation process of video shot S_2 are $\Gamma_2 = \Gamma_2 \cup \Gamma_{H_1}$. Similarly, the considered assertions for the video shot S_i are defined as follows:

$$\Gamma_i = \Gamma_i \cup \Gamma_{H_{i-1}} \cup \ldots \cup \Gamma_{H_1} \tag{13}$$

In other words, the high level concept assertions generated for each video shot will be considered for the interpretation process of the next video shots. Taking the assertions of the previous video shots to the next video shots is performed only for the high level concept assertions and not for the low level concept assertions.

3.4 New Results on Optimization Techniques

In the RMI architecture procedure for probabilistic queries there is interest in representing both probabilistic and deterministic dependencies. We have introduced a Markovian style of probabilistic reasoning in first-order logic known as Markov logic and have investigated the opportunities for restricting this formalism to DLs.

In [Gries and Möller, 2010] we show that Gibbs sampling with deterministic dependencies specified in an appropriate fragment remains correct, i.e., probability estimates approximate the correct probabilities. We have investigated a Gibbs sampling method incorporating deterministic dependencies and conclude that this incorporation can speed up Gibbs sampling significantly. For details see [Gries and Möller, 2010].

4 Complete Example

One of the main innovation introduced in the previous section, namely the introduction of a probabilistic preference measure to control the space of possible interpretations, is exemplified here using examples inspired by the environmental domain used in CASAM.

At the beginning of this example, the **signature** of the knowledge base is presented. The set of all concept names **CN** is divided into two different sets **Events** and **PhysicalThings** such that

$$\mathbf{CN} = \mathbf{Events} \cup \mathbf{PhysicalThings} \tag{14}$$

where these two sets are defined as follows:

$$\mathbf{Events} = \{CarEntry, EnvConference, EnvProt, HealthProt\} \tag{15}$$
$$\mathbf{PhysicalThings} = \{Car, DoorSlam, Building, Environment, Agency\} \tag{16}$$

EnvConference, *EnvProt* and *HealthProt* denote environmental conference, environmental protection and health protection, respectively.

The set of role names **RN** is defined as follows:

$$\mathbf{RN} = \{Causes, OccursAt, HasAgency, HasTopic, HasSubject, HasObject, HasEffect, HasSubEvent, HasLocation\}$$

In the following, the set of individual names **IN** is given by

$$\mathbf{IN} = \{C_1, DS_1, ES_1, Ind_{42}, Ind_{43}, Ind_{44}, Ind_{45}, Ind_{46}, Ind_{47}, Ind_{48}\} \tag{17}$$

Note that the notations in this example are based on Alchemy notations i.e. the instance-, concept- and role names begin with capital letters. In the next table, the set of forward chaining rules $\mathcal{FR}$ is defined:

$\forall x$	$CarEntry(x)$	$\rightarrow$	$\exists y \; Building(y), OccursAt(x, y)$
$\forall x$	$EnvConference(x)$	$\rightarrow$	$\exists y \; Environment(y), HasTopic(x, y)$
$\forall x$	$EnvProt(x)$	$\rightarrow$	$\exists y \; Agency(y), HasAgency(x, y)$

Table 1: Set of forward chaining rules $\mathcal{FR}$

Similarly, the set of backward chaining rules $\mathcal{BR}$ is depicted as follows:

$Causes(x, y)$	$\leftarrow$	$CarEntry(z), HasObject(z, x), HasEffect(z, y), Car(x), DoorSlam(y)$
$OccursAt(x, y)$	$\leftarrow$	$EnvConference(z), HasSubEvent(z, x), HasLocation(z, y), CarEntry(x), Building(y)$
$HasTopic(x, y)$	$\leftarrow$	$EnvProt(z), HasSubEvent(z, x), HasObject(z, y), EnvConference(x), Environment(y)$
$HasAgency(x, y)$	$\leftarrow$	$HealthProt(z), HasObject(z, x), HasSubject(z, y), EnvProt(x), Agency(y)$

Table 2: Set of backward chaining rules $\mathcal{BR}$

In the following a set of weighted rules $\mathcal{WR}$ is given where all rules have the same high weight set to 5. The syntax of each rule is defined as follows: on the left-hand side of each rule an **Events** concept name is given whose variable z is universally quantified whereas on the right-hand side of each rule, the variables x and y are existentially restricted:

5	$\forall z \; CarEntry(z)$	$\Rightarrow$	$\exists x, y$	$Car(x) \wedge DoorSlam(y) \wedge Causes(x, y) \wedge HasObject(z, x) \wedge HasEffect(z, y)$
5	$\forall z \; EnvConference(z)$	$\Rightarrow$	$\exists x, y$	$CarEntry(x) \wedge Building(y) \wedge OccursAt(x, y) \wedge HasSubEvent(z, x) \wedge HasLocation(z, y)$
5	$\forall z \; EnvProt(z)$	$\Rightarrow$	$\exists x, y$	$EnvConference(x) \wedge Environment(y) \wedge HasTopic(x, y) \wedge HasSubEvent(z, x) \wedge HasObject(z, y)$
5	$\forall z \; HealthProt(z)$	$\Rightarrow$	$\exists x, y$	$EnvProt(x) \wedge Agency(y) \wedge HasAgency(x, y) \wedge HasObject(z, x) \wedge HasSubject(z, y)$

Table 3: Set of weighted rules $\mathcal{WR}$

Note that the weighted rules $\mathcal{WR}$ and their weights can be learnt by the machine learning component of the CASAM project.

The selected initial value for ϵ in this example is 0.05. In the following, Δ_1 and Δ_2 denote respectively the set of assertions hypothesized by a backward chaing rule and the set of assertions generated by a forward chaining rule at each interpretation level.

Let us assume that RMI receives the following weighted Abox $\mathcal{A}$ from the KDMA and HCI components:

1.3	$Car(C_1)$
1.2	$DoorSlam(DS_1)$
-0.3	$EngineSound(ES_1)$
	$Causes(C_1, DS_1).$

Table 4: Received Abox $\mathcal{A}$ from KDMA and HCI

where the strict assertions are terminated by a period. The first applied operation to A is the MAP function which returns the next bit vector W:

Ground atoms	W
$Car(C_1)$	1
$DoorSlam(DS_1)$	1
$EngineSound(ES_1)$	0
$Causes(C_1, DS_1)$	1

Table 5: Ground atoms and bit vector W

The vector W is composed of positive and negative events (bits). By applying the *Select* function to W_k and the input Abox, the assertions from the input Abox are selected that correspond to positive events in W_k. Additionally, the assigned weights to the positive assertions are taken from the input Abox $\mathcal{A}$. In the following, Abox A_0 is depicted which contains the positive assertions:

1.3	$Car(C_1)$
1.2	$DoorSlam(DS_1)$
	$Causes(C_1, DS_1).$

Table 6: Abox $\mathcal{A}_0$, the result of *Select* function

The P function determines the full joint probability of the observations chosen by the *Select* function:

$$p_0 = P(Car(C_1) \wedge DoorSlam(DS_1) \wedge Causes(C_1, DS_1)) = 0.650 \tag{18}$$

Since no appropriate forward chaining rule from $\mathcal{FR}$ is applicable to Abox $\mathcal{A}_0$,

$$forward\_chain(\mathcal{T}, \mathcal{FR}, \mathcal{A}_0) = \emptyset \tag{19}$$

and as a result $\mathcal{A}_0 = \mathcal{A}_0 \cup \emptyset$. The next step is the performance of *backward_chain* function where the next backward chaining rule from $\mathcal{BR}$ can be applied to Abox $\mathcal{A}_0$:

$$Causes(x, y) \leftarrow CarEntry(z), HasObject(z, x), HasEffect(z, y), Car(x), DoorSlam(y) \tag{20}$$

Consequently, by applying the above rule the next set of assertions is hypothesized:

$\Delta_1 = \{CarEntry(Ind_{42}), HasObject(Ind_{42}, C_1), HasEffect(Ind_{42}, DS_1)\}$

which are considered as strict assertions. In the following, $A_1 = A_0 \cup \Delta_1$ is depicted:

1.3	$Car(C_1)$
1.2	$DoorSlam(DS_1)$
	$Causes(C_1, DS_1)$.
	$CarEntry(Ind_{42})$.
	$HasObject(Ind_{42}, C_1)$.
	$HasEffect(Ind_{42}, DS_1)$.

Table 7: Abox A_1

In the above Abox, the full joint probability of observations is determined again:

$$p_1 = P(Car(C_1) \wedge DoorSlam(DS_1) \wedge Causes(C_1, DS_1)) = 0.840 \tag{21}$$

As it can be seen, $p_1 > p_0$ i.e. $P(\mathcal{A}_0, \mathcal{A}_i, \mathcal{WR}, \mathcal{T})$ increases by adding the new hypothesized assertions. This shows that the new assertions are considered as additional support. The termination condition of the algorithm is not fulfilled therefore the algorithm continues processing. At this level, it is still not known whether Abox $\mathcal{A}_1$ can be considered as the final interpretation Abox. Thus, this process is continued with another level. Consider the next forward chaining rule:

$$\forall x \;\; CarEntry(x) \rightarrow \exists y \;\; Building(y), OccursAt(x, y) \tag{22}$$

By applying the above rule, the next set of assertions is generated namely:

$$\Delta_2 = \{Building(Ind_{43}), OccursAt(Ind_{42}, Ind_{43})\} \tag{23}$$

The new generated asssertions are also considered as strict assertions. In the following, the expanded Abox $\mathcal{A}_1 = \mathcal{A}_1 \cup \Delta_2$ is illustrated:

1.3	$Car(C_1)$
1.2	$DoorSlam(DS_1)$
	$Causes(C_1, DS_1)$.
	$CarEntry(Ind_{42})$.
	$HasObject(Ind_{42}, C_1)$.
	$HasEffect(Ind_{42}, DS_1)$.
	$Building(Ind_{43})$.
	$OccursAt(Ind_{42}, Ind_{43})$.

Table 8: Expanded Abox $\mathcal{A}_1$

Let us assume the next backward chaining rule:

$$OccursAt(x, y) \;\leftarrow\; EnvConference(z), HasSubEvent(z, x), HasLocation(z, y), CarEntry(x), Building(y)$$

Consequently, by applying the above abduction rule the next set of new assertions is hypothesized:

$$\Delta_1 = \{EnvConference(Ind_{44}), HasSubEvent(Ind_{44}, Ind_{42}), HasLocation(Ind_{44}, Ind_{43})\}$$

which are considered as strict assertions. In the following, $A_2 = A_1 \cup \Delta_1$ is depicted:

1.3	$Car(C_1)$
1.2	$DoorSlam(DS_1)$
	$Causes(C_1, DS_1).$
	$CarEntry(Ind_{42}).$
	$HasObject(Ind_{42}, C_1).$
	$HasEffect(Ind_{42}, DS_1).$
	$Building(Ind_{43}).$
	$OccursAt(Ind_{42}, Ind_{43}).$
	$EnvConference(Ind_{44}).$
	$HasSubEvent(Ind_{44}, Ind_{42}).$
	$HasLocation(Ind_{44}, Ind_{43}).$

Table 9: Abox $\mathcal{A}_2$

In the above Abox, again the full joint probability of observations is determined:

$$p_2 = P(Car(C_1) \wedge DoorSlam(DS_1) \wedge Causes(C_1, DS_1)) = 0.819 \tag{24}$$

As it can be seen, $p_2 < p_1$ i.e. the full joint probability of the observations decreases slightly by adding the new hypothesized assertions. The termination condition of the algorithm is fulfilled. Therefore Abox $\mathcal{A}_1$ can be considered as the output Abox. To realize how the further behaviour of the probabilities is, this process is continued. Consider the next forward chaining rule:

$$\forall x \;\; EnvConference(x) \rightarrow \exists y \;\; Environment(y), HasTopic(x, y) \tag{25}$$

By applying the above rule, new assertions are generated.

$$\Delta_2 = \{Environment(Ind_{45}), HasTopic(Ind_{44}, Ind_{45})\} \tag{26}$$

In the following, the expanded Abox $\mathcal{A}_2 = \mathcal{A}_2 \cup \Delta_2$ is depicted:

1.3	$Car(C_1)$
1.2	$DoorSlam(DS_1)$
	$Causes(C_1, DS_1).$
	$CarEntry(Ind_{42}).$
	$HasObject(Ind_{42}, C_1).$
	$HasEffect(Ind_{42}, DS_1).$
	$Building(Ind_{43}).$
	$OccursAt(Ind_{42}, Ind_{43}).$
	$EnvConference(Ind_{44}).$
	$HasSubEvent(Ind_{44}, Ind_{42}).$
	$HasLocation(Ind_{44}, Ind_{43}).$
	$Environment(Ind_{45}).$
	$HasTopic(Ind_{44}, Ind_{45}).$

Table 10: Expanded Abox $\mathcal{A}_2$

Consider the next backward chaining rule:

$HasTopic(x, y) \leftarrow EnvProt(z), HasSubEvent(z, x), HasObject(z, y), EnvConference(x), Environment(y)$

By applying the above abduction rule, the following set of assertions is hypothesized:

$\Delta_1 = \{EnvProt(Ind_{46}), HasSubEvent(Ind_{46}, Ind_{44}), HasObject(Ind_{46}, Ind_{45})\}$

which are considered as strict assertions. In the following, $\mathcal{A}_3$ is depicted so that $A_3 = A_2 \cup \Delta_1$:

1.3	$Car(C_1)$
1.2	$DoorSlam(DS_1)$
	$Causes(C_1, DS_1).$
	$CarEntry(Ind_{42}).$
	$HasObject(Ind_{42}, C_1).$
	$HasEffect(Ind_{42}, DS_1).$
	$Building(Ind_{43}).$
	$OccursAt(Ind_{42}, Ind_{43}).$
	$EnvConference(Ind_{44}).$
	$HasSubEvent(Ind_{44}, Ind_{42}).$
	$HasLocation(Ind_{44}, Ind_{43}).$
	$Environment(Ind_{45}).$
	$HasTopic(Ind_{44}, Ind_{45}).$
	$EnvProt(Ind_{46}).$
	$HasSubEvent(Ind_{46}, Ind_{44}).$
	$HasObject(Ind_{46}, Ind_{45}).$

Table 11: Abox $\mathcal{A}_3$

In the above Abox $\mathcal{A}_3$, again the full joint probability of observations is determined:

$$p_3 = P(Car(C_1) \land DoorSlam(DS_1) \land Causes(C_1, DS_1)) = 0.833 \tag{27}$$

As it can be seen, $p_3 > p_2$, i.e. the full joint probability of the observations increases slightly by adding the new hypothesized assertions.

Consider the next forward chaining rule:

$$\forall x \quad EnvProt(x) \rightarrow \exists y \quad Agency(y), HasAgency(x, y) \tag{28}$$

By applying the above rule, the next assertions are generated:

$$\Delta_2 = \{Agency(Ind_{47}), HasAgency(Ind_{46}, Ind_{47})\} \tag{29}$$

In the following the expanded Abox $\mathcal{A}_3 = \mathcal{A}_3 \cup \Delta_2$ is illustrated:

1.3	$Car(C_1)$
1.2	$DoorSlam(DS_1)$
	$Causes(C_1, DS_1)$.
	$CarEntry(Ind_{42})$.
	$HasObject(Ind_{42}, C_1)$.
	$HasEffect(Ind_{42}, DS_1)$.
	$Building(Ind_{43})$.
	$OccursAt(Ind_{42}, Ind_{43})$.
	$EnvConference(Ind_{44})$.
	$HasSubEvent(Ind_{44}, Ind_{42})$.
	$HasLocation(Ind_{44}, Ind_{43})$.
	$Environment(Ind_{45})$.
	$HasTopic(Ind_{44}, Ind_{45})$.
	$EnvProt(Ind_{46})$.
	$HasSubEvent(Ind_{46}, Ind_{44})$.
	$HasObject(Ind_{46}, Ind_{45})$.
	$Agency(Ind_{47})$.
	$HasAgency(Ind_{46}, Ind_{47})$.

Table 12: Expanded Abox $\mathcal{A}_3$

Let us consider the next backward chaining rule:

$HasAgency(x, y) \leftarrow HealthProt(z), HasObject(z, x), HasSubject(z, y), EnvProt(x), Agency(y)$

Consequently, new assertions are hypothesized by applying the above abduction rule, namely:

$\Delta_1 = \{HealthProt(Ind_{48}), HasObject(Ind_{48}, Ind_{46}), HasSubject(Ind_{48}, Ind_{47})\}$

which are considered as strict assertions. In the following, $\mathcal{A}_4$ is depicted so that $A_4 = A_3 \cup \Delta_1$:

1.3	$Car(C_1)$
1.2	$DoorSlam(DS_1)$
	$Causes(C_1, DS_1).$
	$CarEntry(Ind_{42}).$
	$HasObject(Ind_{42}, C_1).$
	$HasEffect(Ind_{42}, DS_1).$
	$Building(Ind_{43}).$
	$OccursAt(Ind_{42}, Ind_{43}).$
	$EnvConference(Ind_{44}).$
	$HasSubEvent(Ind_{44}, Ind_{42}).$
	$HasLocation(Ind_{44}, Ind_{43}).$
	$Environment(Ind_{45}).$
	$HasTopic(Ind_{44}, Ind_{45}).$
	$EnvProt(Ind_{46}).$
	$HasSubEvent(Ind_{46}, Ind_{44}).$
	$HasObject(Ind_{46}, Ind_{45}).$
	$Agency(Ind_{47}).$
	$HasAgency(Ind_{46}, Ind_{47}).$
	$HealthProt(Ind_{48}).$
	$HasObject(Ind_{48}, Ind_{46}).$
	$HasSubject(Ind_{48}, Ind_{47}).$

Table 13: Abox $\mathcal{A}_4$

In the above Abox, again the full joint probability of observations is determined:

$$p_4 = P(Car(C_1) \wedge DoorSlam(DS_1) \wedge Causes(C_1, DS_1)) = 0.837 \tag{30}$$

As it can be seen, $p_4 > p_3$, i.e. the full joint probability of the observations increases slightly by adding the new hypothesized assertions.

Evaluation of the Results

The determined probability values $P(\mathcal{A}_0, \mathcal{A}_i, \mathcal{R}, \mathcal{WR}, \mathcal{T})$ of this example are summarized in the next table which shows clearly the behaviour of the probabilities stepwise after perfoming the interpretation process:

i	*Abox* $\mathcal{A}_i$	$p_i = P(\mathcal{A}_0, \mathcal{A}_i, \mathcal{WR}, \mathcal{T})$
0	$\mathcal{A}_0$	$p_0 = 0.650$
1	$\mathcal{A}_1$	$p_1 = 0.840$
2	$\mathcal{A}_2$	$p_2 = 0.819$
3	$\mathcal{A}_3$	$p_3 = 0.833$
4	$\mathcal{A}_4$	$p_4 = 0.837$

Table 14: Summary of the probability values

Variable i denotes in the above table the successive levels of the interpretation process. In this example, the interpretation process is consecutively performed four times. As it can be seen in Table 14, through the first interpretation level the probability p_1 increases strongly in comparison to p_0. By performing the second, third and the forth interpretation levels, the probability values decrease slightly in comparison to p_1. This means no significant changes can be seen in the results. In other words, the determination of $\mathcal{A}_3$ and $\mathcal{A}_4$ were not required at all. But the determination of $\mathcal{A}_2$ was required to realize the slight difference $|p_2 - p_1| < \frac{\epsilon}{2}$. Consequently, Abox $\mathcal{A}_1$ is considered as the final interpretation Abox.

At the end of this example, we have to mention that this example was not constructed to show the possible branchings through the interpretation process. But the purpose of this example was to show how the probabilities of the most probable world of observations $P(\mathcal{A}_0, \mathcal{A}, \mathcal{R}, \mathcal{WR}, \mathcal{T})$ behave during the interpretation process if there is only one possible branch to follow.

6 Conclusion and Remarks

For multimedia interpretation, a semantically well-founded formalization is required. In accordance with previous work, in CASAM a well-founded abduction-based approach is pursued. Extending previous work, abduction is controlled by probabilistic knowledge, and it is done in terms of first-order logic.

This report describes the probabilistic abduction engine and the optimization techniques for multimedia interpretation. It extends deliverable D3.2 by providing a probabilistic scoring function for ranking interpretation alternatives. Parameters for the CASAM Abduction Engine (CAE) introduced already in D3.2 are now appropriately formalized first the first such that CAE is better integrated into the probabilistic framework.

In addition, this deliverable describes how media interpretation services can be provided that work incrementally, i.e., are able to consume new analysis results, or new input from a human annotator, and produce notifications for additional interpretation results or, in some cases, revision descriptions for previous interpretations. Incremental processing is nontrivial and is realized using an Abox difference operator, which is used to interpretation results obtained for extended inputs with one(s) previously obtained such that notifications about additions and revisions can be computed.

References

[Baader et al., 2003] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press.

[Castano et al., 2008] Castano, S., Espinosa, S., Ferrara, A., Karkaletsis, V., Kaya, A., Möller, R., Montanelli, S., Petasis, G., and Wessel, M. (2008). Multimedia interpretation for dynamic ontology evolution. In *Journal of Logic and Computation.* Oxford University Press.

[Domingos and Richardson, 2007] Domingos, P. and Richardson, M. (2007). Markov logic: A unifying framework for statistical relational learning. In Getoor, L. and Taskar, B., editors, *Introduction to Statistical Relational Learning*, pages 339–371. Cambridge, MA: MIT Press.

[Gries and Möller, 2010] Gries, O. and Möller, R. (2010). Gibbs sampling in probabilistic description logics with deterministic dependencies. In *Proc. of the First International Workshop on Uncertainty in Description Logics, Edinburgh.*

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA.