# Formalization of the CRM: Initial Thoughts

Carlo Meghini

Istituto di Scienza e Tecnologie della Informazione
Consiglio Nazionale delle Ricerche – Pisa

CRM SIG Meeting
Iraklio, October 1st, 2014

# Outline

- Descriptions, Sentences and Their Names
- Knowing that

# Descriptions, Sentences and Their Names

Documentation KB:

- *Domain* knowledge: descriptions of the individuals in the domain, sentences on these descriptions.
  Used by the end-users, for instance to discover and access the resources of the library
- *Documentation* knowledge: descriptions of domain knowledge.
  Used by digital librarians in order to manage the resources of the library:
  - provenance
  - degree of trust
  - the access policy

  of a metadata record.

# Documentation knowledge

Documentation knowledge consists of factual and ontological statements about individuals, concepts and relations of the domain of dicourse.

It can be expressed and reasoned upon using standard tools, such as OWL.

However, a problem arises if documentation knowledge is to be used *together with* domain knowledge, as it happens in digital libraries.

In order to express knowledge about resources and about the statements used to describe such resources *in the same language,* one needs very powerful languages, whose expressive power goes beyond that of first-order logic.

Such languages, though, are hardly usable in digital libraries, because their negative computational properties are inadequate to digital library requirements.

# RDF can do it!

RDF allows to express metadata records as statements having the described resource as subject and metadata elements as properties.

It also allows to express a certain amount of documentation knowledge, by allowing properties as subjects in statements.

However, as soon as the expressivity of the language goes beyond that of RDF Schema by including constructs from Description Logics, serious computational problems arise.

Indeed, the combination of RDF with Description Logics yields the language OWL Full, which is undecidable in spite of the decidability of its two constituents.

# A different approach

*Simultaneous* usage of two different logics:

- the *object* logic devoted to represent domain knowledge
- *documentation* logic devoted to represent documentation knwoledge.

With these two logics we build a *documentation system*, which has three components:

1. the domain knowledge base
2. the documentation knowledge base
3. the *naming function* that associates the individual constants (*i.e.*, the names) in the documentation knowledge base with the metadata records (*i.e.*, the sentences) in the domain knowledge base.

*Documentation systems* are the backbones of digital libraries.

# New discovery functionality

The two KBs in a DS can be queried individually, based on the kind of knwoledge they store.

*object-query:* $\textsc{Ask}_o(\mathit{Book} \sqcap \exists \mathit{Title}.\{\mathit{Waverley}\})$.

*doc-query:* $\textsc{Ask}_d(\exists \mathsf{CAss}.\{b\})$ asking for the metadata records about book $b$.

*mixed-object* queries, asking for the resourses of the object-KB that satisfy some property expressed in the doc-KB; for instance, the editions of the *Waverley* that have been described by John

$$\mathit{Book} \sqcap \exists \mathit{Title}.\{\mathit{Waverley}\} \sqcap \textsc{Ask}_d(\exists \mathsf{CAss}^{-}.\textsc{Ask}_o(\exists \mathit{Author}.\{\mathit{John}\}))$$

*mixed-doc* queries, asking for the resourses of the doc-KB that satisfy some property expressed in the object-KB; for instance, the metadata records of the *Waverley* that have been created by Sue.

$$\exists \mathsf{MRD}.(\exists \mathsf{CAss}.\textsc{Ask}_o(\exists \mathit{Title}.\{\mathit{Waverley}\})) \sqcap \textsc{Ask}_o(\exists \mathit{Author}.\{\mathit{Sue}\})$$

Maybe the same separation of concerns applies to the CRM.

In the CRM we have classes and properties that can be used for expressing domain knowledge:

- E24 Physical Man-Made Thing, P46 is composed of

and classes and properties that can be used for expressing documentation knowledge:

- E90 Symbolic Object, P140 assigned attribute to

Would we get a more readable KB if we separated documentation from domain knowledge?

# Knowing that

*Knowledge* is a relation between a knower, someone endowed with mind, and a proposition, that is, the idea expressed by a simple declarative sentence.

In this sense, knowledge is a *propositional attitude*.

In order to be able to talk about of what is known or not known we augment the language so that for every sentence $\alpha$ there is another sentence $\mathbf{K}\alpha$ that can be read as "$\alpha$ is known".

- "$\alpha$ is *known* to be true" *is expressed as* "$\alpha$ is known" is true: $\mathbf{K}\alpha$
- "$\alpha$ is *not known* to be true" *is expressed as* "$\alpha$ is known" is false: $\neg\mathbf{K}\alpha$

In this way, we only have to talk about which sentences are true or false, the same way we talk about the world.

The language obtained by adding to $\mathcal{L}$ sentences of the form $\mathbf{K}\alpha$ is $\mathcal{KL}$.

But what is the semantics of a sentence like $\mathbf{K}\alpha$?

At any particular time, the knower will not have determined the actual world state in full detail, but perhaps some possibilities will have been ruled out. As more information is acquired, more possibilities are ruled out, until maybe only one world state remains.

Until this single-world state is reached, the knower retains several world states as possible, meaning that the knower supposes that the actual world is one of these, even though it is not determined which in particular.

An *epistemic state* of a knower can therefore be modelled as a set $S$ of world states, all of which are retained as possible by the knower.

What does a knower knows, in this situation?

According to Hintikka, what is known for sure is exactly what is true in *all* the world states in the epistemic state.

As a consequence, $\mathbf{K}\alpha$ is true in a particular epistemic state if $\alpha$ is true in all the world states that are part of the current epistemic state $S$.

To have a *complete* knowledge of the world means to be in an epistemic state consisting of a single world state $\{w\}$.

Under this circumstance, for every sentence of $\mathcal{L}$ the knower is able to state whether or not $\mathbf{K}\alpha$ is true.

The problem is that the knower may be wrong, in the sense that *the actual world state may not be in the epistemic state of the knower,* even though the knower supposes so. In this case, we say that the knowledge is not *accurate.*

In order to model the accuracy of the knowledge, we add to the epistemic state the indication of which world state is the real one.

So now an epistemic state is a pair $(S, w)$.

This addition allows us to determine whether our knowledge is accurate. In particular, in an epistemic state $(S, w)$,

- if the real world state $w$ is in the epistemic state $S$, then what is known is also true in reality, and our knowledge is accurate.
- If the real real world state $w$ is not in the epistemic state $S$, then what is known may not be true in reality, and our knowledge may not be accurate.

To have a *complete and accurate* knowledge of the world means to be in the epistemic state $(\{w\}, w)$, consisting of the real world state.

# Knowledge and truth

There are sentences that are true but not known:

- $\alpha \wedge \neg \mathbf{K}\alpha$ is satisfiable

There are sentences that are false and known

- $\neg\alpha \wedge \mathbf{K}\alpha$ is satisfiable

Therefore it is not generally true that a true sentence is known:

- $\not\models (\alpha \supset \mathbf{K}\alpha)$
- $\not\models (\mathbf{K}\alpha \supset \alpha)$

Valid sentences are always known

- $\models \alpha$ then $\models \mathbf{K}\alpha$

# ASK and TELL operations

Now that we have a language for talking about knowledge, we use it to interact with a KB. The interaction is based on three operations:

1. $\mathit{Ask}[\alpha, e] \in \mathit{yes}, \mathit{no}$
   In an epistemic state $e$ we determine whether $\alpha$ is known by asking it and getting a *yes* or a *no.*
2. $\mathit{Tell}[\alpha, e] = e'$
   In an epistemic state $e$ we add $\alpha$ by telling it, and getting the system in a new epistemic state $e'$.
3. $\mathbf{INITIAL}[\,] = e_0$
   where $e_0$ is the epistemic state before any *Tell* operation. The initial state $e_0$ includes all world states, since at the beginning nothing is known, except for the valid sentences.

The idea is that we imagine the lifetime of a KB as proceeding through a number of states $e_0, e_1, e_2, \ldots$ such that for every $i > 0$, $e_i = \mathit{Tell}[\alpha_i, e_{i-1}]$. In any such state, we can use *Ask* to determine what is known.

# Ask

The purpose of *Ask* is to enable a user to find out from a KB whether a sentence $\alpha$ is true or not. There can be four possible outcomes:

1. the KB may believe that $\alpha$ is true ($\mathbf{K}\alpha \wedge \neg\mathbf{K}\neg\alpha$ is satisfiable)
2. the KB may believe that $\alpha$ is false ($\mathbf{K}\neg\alpha \wedge \neg\mathbf{K}\alpha$ is satisfiable)
3. the KB may not know if $\alpha$ is true or false ($\neg\mathbf{K}\alpha \wedge \neg\mathbf{K}\neg\alpha$ is satisfiable)
4. the KB may be in an inconsistent state and believe everything ($\mathbf{K}\alpha \wedge \mathbf{K}\neg\alpha$ is satisfiable)

In order to find out which is the case, we can define ask as follows:

$$Ask[\alpha, e] = \begin{cases} yes & \text{if } e \models \mathbf{K}\alpha \\ no & \text{otherwise} \end{cases}$$

In this way, to find out which of the four above possibilities is the case, we can first ask $\alpha$ and then $\neg\alpha$. With two interactions we have obtained what we wanted.

# Provisional Conclusions

How do we make knower out of a CRM KB?