LoE – the Logic of Entities

Oct 13,2023

LoE – The Logic of entities

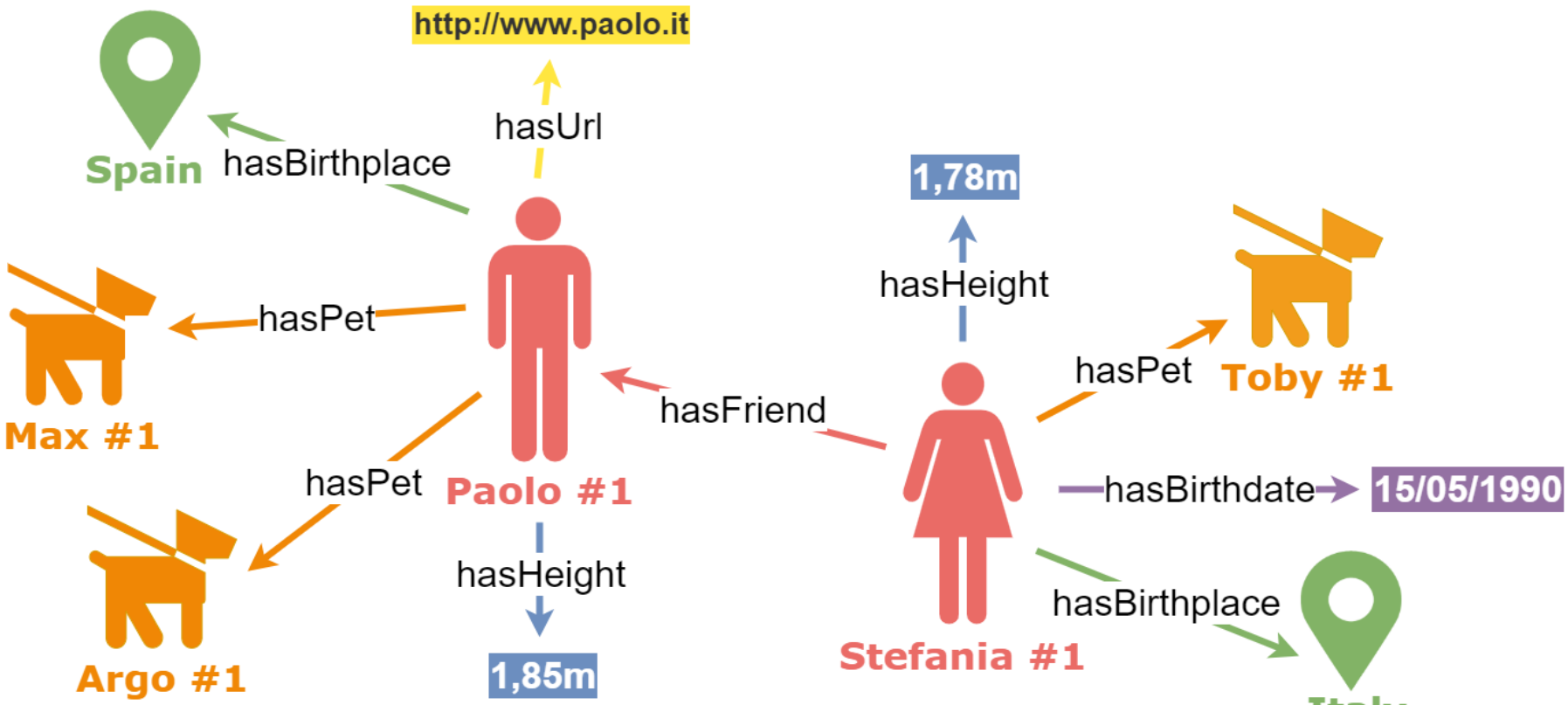
- The **Logic of Entities (**LOE**)** allows us to describe and reason about entities as a function of the main class to which they belong, and their properties.
- All entities are assumed to have a name which uniquely identifies them, which may be unknown.
- LOE is the simplest logic allowing to reason with Entity Graphs (EGs).
- It is KG world model
- It is conceptually similar in spirit to an earlier logic called the Assertion Box (ABOX) of Description Logics (DL).

LoE – The Logic of entities

LOE support the following Entity Graph (EG) fact elements:

- An **Entity** is anything to which we give a name;
- A **Concept** is the main class to which an entity belongs;
- A **(Data) Value** is anything which has a (predefined) name;
- A **Datatype** is a class of (data) values;
- A **Data Property**, also called **Attribute**, is a characteristic of an entity;
- An **Object Property** describes a relation between two entities.

An example of EG



LoE – The Logic of entities - definition

Definition (LoE)

$$LOE = \langle EG, \models_{LOE} \rangle$$

with

$$EG = \langle L_{LOE}, D_{LOE}, I_{LOE} \rangle$$

When no confusion arises, below we drop the subscripts.

LoE – Domain/facts

Definition (Domain, intensional definition)

$$D_i = < E, \{C\}, \{R\} >$$

where:

$$E = \{e\} \cup \{v\}$$
$$\{C\} = ET \cup DT$$
$$\{R\} = \{OR\} \cup \{DR\}$$

where E is a set of **entities** and **values**, ET = $\{E_T\}$, $E_T = \{e\}$ and DT = $\{D_T\}$, $D_T = \{v\}$ are **sets of entity types (etypes)** and **data types** (**dtypes)**, respectively, and OR, DR are **(binary) object** and **data relations**.

Observation. LOE allows for the following facts:

- Every etype/ dtype and its argument is a fact.
- Every relation R and its two arguments is a fact.

Facts only have one of for possible forms: $E_T(e)$, $D_T(v)$ and $OR(e_i, e_j)$, DR(e, v).

etypes - example

An example of etype is: *Location*, i.e., an etype which models spatial containment of entities.

Locations do not change their position with respect to their coordinate reference systems. Their space coordinates are therefore an important proxy for deciding whether two locations (i.e., two entities belonging to the etype Location) are actually the same location.

There are many etypes which are special cases (sub-etypes) of Location, for instance: *Mountain*, *City*, *Street*, *Home*, and many others.

Other important etypes are:

- *Entity*, the most general etype, that which contains all elements in ET. (its most relevant property is that it has a name, thus imposing that all entities must have a name);
- *Event*, whose most characterizing properties are its start and end times,
- *Person*, with properties such as name, birth date, and parents; and many others.

dtypes

Observation 7.2. Dtypes have the same properties as etypes plus two more:

- the set of their members, i.e., their values, is predefined and
- the names of values are the same as the values themselves (that is data values denote themselves, thus for instance the number (properly called a numeral) 3.14 is the name of the number 3.14).

Example 7.2 (Dtype) The following is a not exhaustive list of datatypes:

dtype, Float, Integer, Boolean, String, SpaceTime, Identifier

where: sub-dtypes of SpaceTime are GeoCoordinate, Distance, XYCoordinate but also Date, Time, DateTime, and so on.

dtype is the set of all the data values.

etypes and dtypes - observation

Observation (Etype, dtype) In a KG, E is structured into a set of sub-universes, i.e., etypes and dtypes. In abstract, each such sub-universe is just like a class C ∈ {C}, namely a subset of E.

etypes and dtypes are defined in the language and are application independent.

dtypes (as with programming languages) (etypes very rarely) come with certain type operators builtin, most noticeably:

- a set of constructors which allow to build the elements of a type,
- a recognizer able to determine whether a certain element belongs to a certain type,
- and an equivalence relation which allows to decide whether two elements of that time are the same.

Object and data (binary) relations

Definition (Object and data binary relations {R} **of a** KG**)** The set of relations {R} = {OR} ∪ {DR} is a set of binary relations of a KG such that

$$R \subseteq E_T s \times \{E_T t \cup D_T\}$$

with ETs, ETt ∈ ET and DTt ∈ DT. If R is defined as:

$$OR \subseteq E_T s \times E_T t$$

then we say that OR is a **binary object relation** OR ∈ {OR}. If R is defined as:

$$DR \subseteq E_T \times D_T$$

then we say that DR is an **binary data relation** DR ∈ {DR}.

An example of EG (continued)

E = {#1, #2, #3, #4, #5, #6, #7, #8, #9, 1,85m, 1,78m, 15/05/1990, http://www.paolo.it}

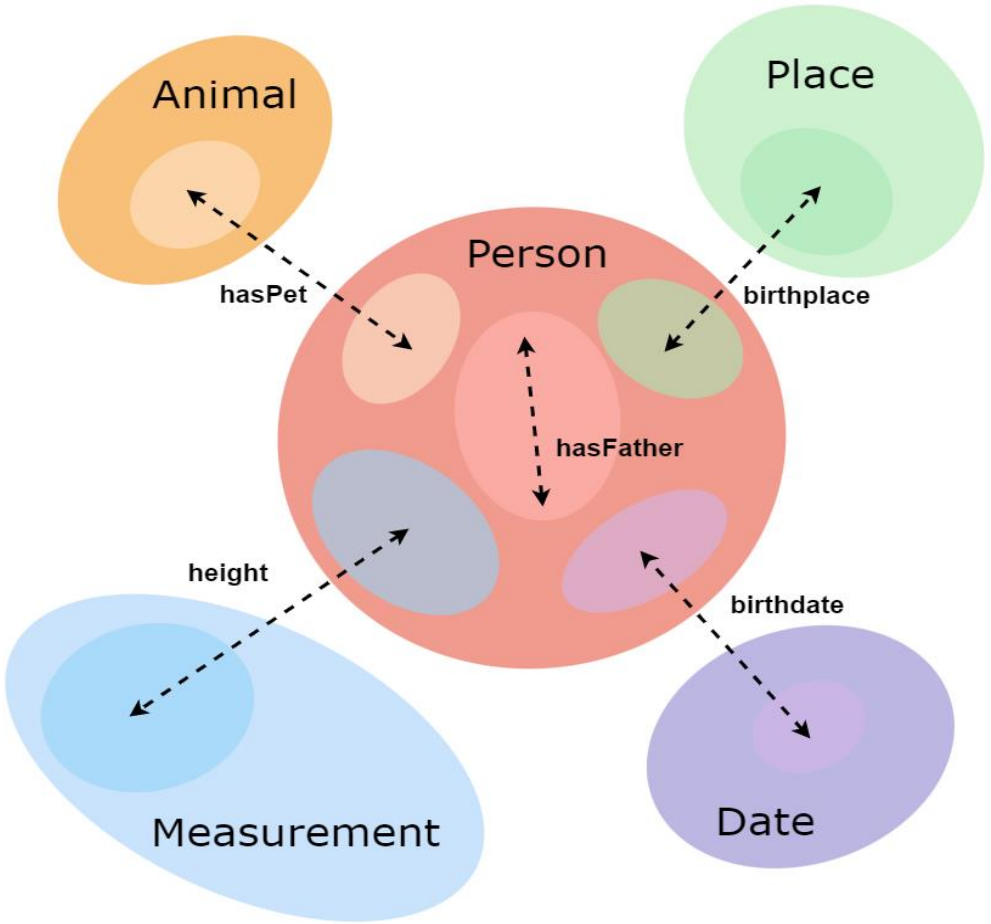
ET = {P, D, L}

DT = {Real, Boolean, String}

{R} = {hF, hD, hH, hB, hL, hU}

Observation (Alphabet and domain components) The cardinality of each of the components of D is bigger than or equal to the cardinality of the corresponding elements of S_e.

An example of EG – Venn diagram (continued)



LoE – Language/wffs (assertions)

Definition 10.6 (Language L**)**

$$L = L_{\mathcal{A}} \cup \emptyset$$

where $L_{\mathcal{A}}$ is a set of assertions. $L_{\mathcal{A}}$ is constructed from an alphabet $A_{\mathcal{a}}$ defined as follows.

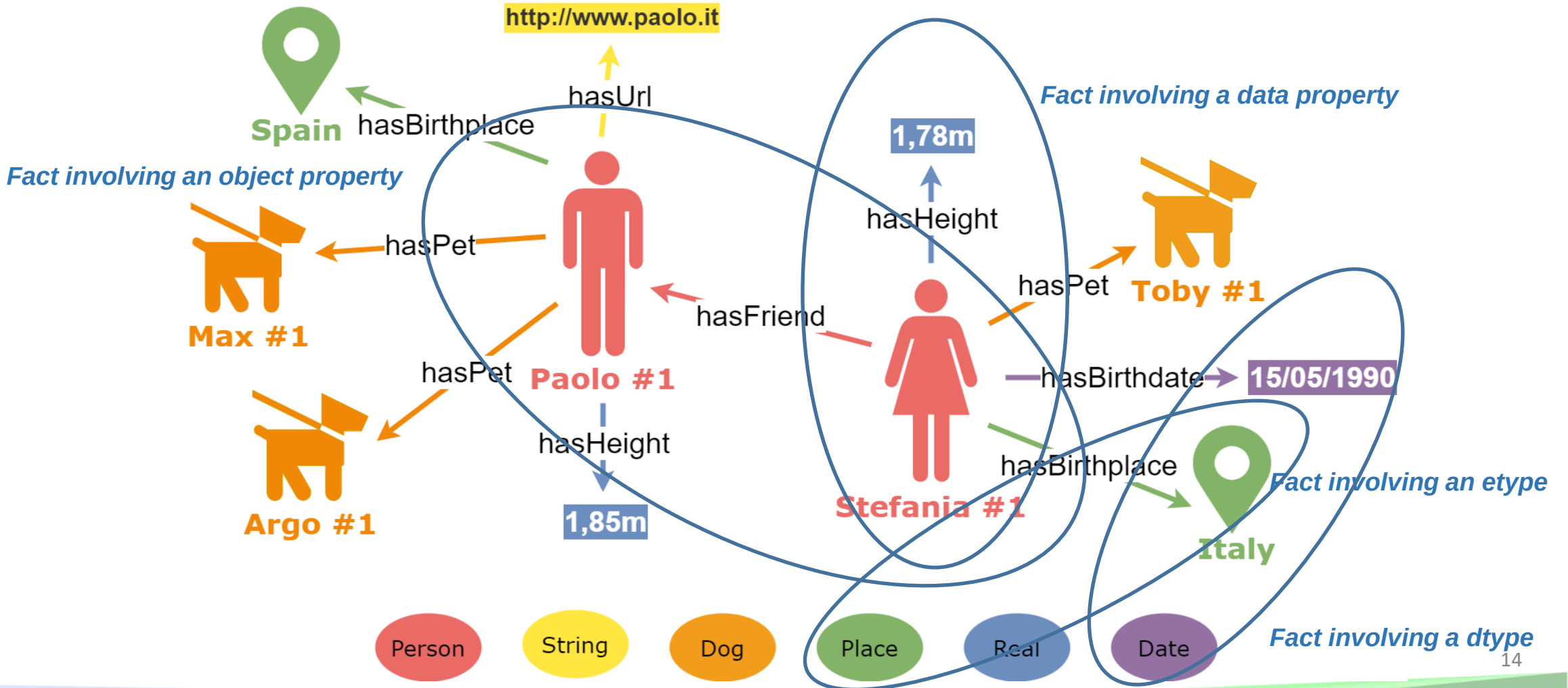
Definition 10.7 (Alphabet A**)**

$$A_{\mathcal{a}} =< E, \{C\}, \{P\} >$$

where E is a set of **(names of) entities** e and of values v, {C}= $ET \cup DT$ is a set of (names of) etypes and of dtypes, {P} is a set of **properties**, also called **roles**, with {P} = {OP} ∪ {DP}, where OP is an **object property** and DP is a **data property**.

$L_{\mathcal{A}}$ is a set of assertions, constructed from the alphabet $A_{\mathcal{a}}$.

An example of EG – how facts are represented in the language



LoE – BNF production rules

<awff> ::= <etype>(<nameEntity>) |
<dtype>(<value>) |
<objProp>(<nameEntity>, <nameEntity>) |
<dataProp>(<nameEntity>, <value>)

<etype> ::= ET1 | . . . | ETn

<dtype> ::= DT1 | . . . | DTn

<objProp> ::= OP1 | . . . | OPn

<dataProp> ::= DP1 | . . . | DPn

<nameEntity> ::= $e1$ | . . . | en

<value> ::= $v1$ | . . . | vn

LoE – Theory (example –as from above)

Alphabet

- *set of entities in* E = {Paolo, Sofia, Stefania, Argo, Max, Toby, Balto, Spain, Italy, Balto Spain, Italy}
- *set of values in* E = {1,85m, 1,78m, 15/05/1990, http://www.paolo.it}
- {P} = {hasFriend, hasDog, hasHeight, hasBirthdate, hasBirthPlace Person, Dog, Place, Measurement, Date}

Assertions = {Person(Paolo), hasBirthplace(Paolo, Spain)
hasHeight(Paolo, 1,85m), hasDog(Paolo, Argo),
hasUrl(Paolo, http://www.paolo.it), hasDog(Paolo, Max)}

etypes and dtypes - observation

Observation (The form of LOE formulas). Assertions have one of four possible forms: *E(e), D(v)* and *OP(ei, ej), DP(e, v)*.

Observation (LOE epressiveness). In LoE, given the low expressiveness of the language, the properties of etypes and dtypes can be exploited to make assertions but they cannot be reasoned about (similar to Relational DBs).

Example. The fact that Fausto is a person cannot be derived from the fact that he is a professor. There is no way to state the subsumption (mono directional definition) that all professors are persons. To complete the KG would require to add this fact to ALL instances of professor.

Interpretation function

Definition (Interpretation function I)

$$I =< I_e, IC, IP >$$

See definition of interpretation function od assertional languages (adjusted to apply to etypes and dtypes

Interpretation function – Example above

We have the following

I(Paolo)	= #1
I(hasFriend(Paolo, Sofia))	= hf(#2, #1)
I(Person(Stefania))	= P(#3)
I(hasDog(Stefania, Toby))	= hD(#3, #6)
I(hasDog(Paolo, Argo))	= hD(#1, #4)
I(hasDog(Paolo, Max))	= hD(#1, #5)

Entailment relation

Definition (Entailment $\models$)

$$M \models w \Longleftrightarrow I(w) \in M$$

with $w \in L$

Reasoning problems

Instance checking, Checking whether an assertion is entailed by a Model, i.e. checking whether

$$M \models E(e)$$
$$M \models P(e1, e2)$$

with M= I(T): A model checking problem!

Reasoning problems

Instance retrieval Given an etype (or object/ data property), retrieve all the entities (or pairs entity, entity/data) which satisfy the etype (object/data property)

$$M \models E$$
$$M \models P$$

with M= I(T): A satisfiability problem!

Observations (The other reasoning problems)

Reasoning Problem (Validity) Given T , check whether for all M, M |= T

NEVER!

Reasoning Problem (Unsatisfiability) Given T , check whether there is no M such that M |= T

NEVER!

Reasoning Problem (Logical consequence) Given T1, T2 and a set of reference models {M}, check whether

$$T1 \models_{\{M\}} T2$$

TRIVIAL! ONLY FOR THE FOMULAS IN T1

Reasoning Problem (Logical equivalence) Given T1, T2 and a set of reference models {M}, check whether

$$T1 \models_{\{M\}} T2 \text{ and } T2 \models_{\{M\}} T1$$

TRIVIAL! T1=T2=T (or subset of T)

Observations (Logical entailment – properties)

Intuition (Reflexivity): $w \models w$

YES!

Intuition (Cut): If $\Gamma \models w1$ and $\Sigma \cup \{w1\} \models w2$ then $\Gamma \cup \Sigma \models w2$

TRIVIAL: only with W2 in Σ

Intuition (Compactness) If $\Gamma \models w$ then there is a finite subset $\Gamma 0 \subseteq \Gamma$ such that $\Gamma 0 \models w$

TRIVIAL: we only have formulas of finite length

Observations (Logical entailment – properties)

Intuition (Monotonicity): If $\Gamma \models w$ then $\Gamma \cup \Sigma \models w$

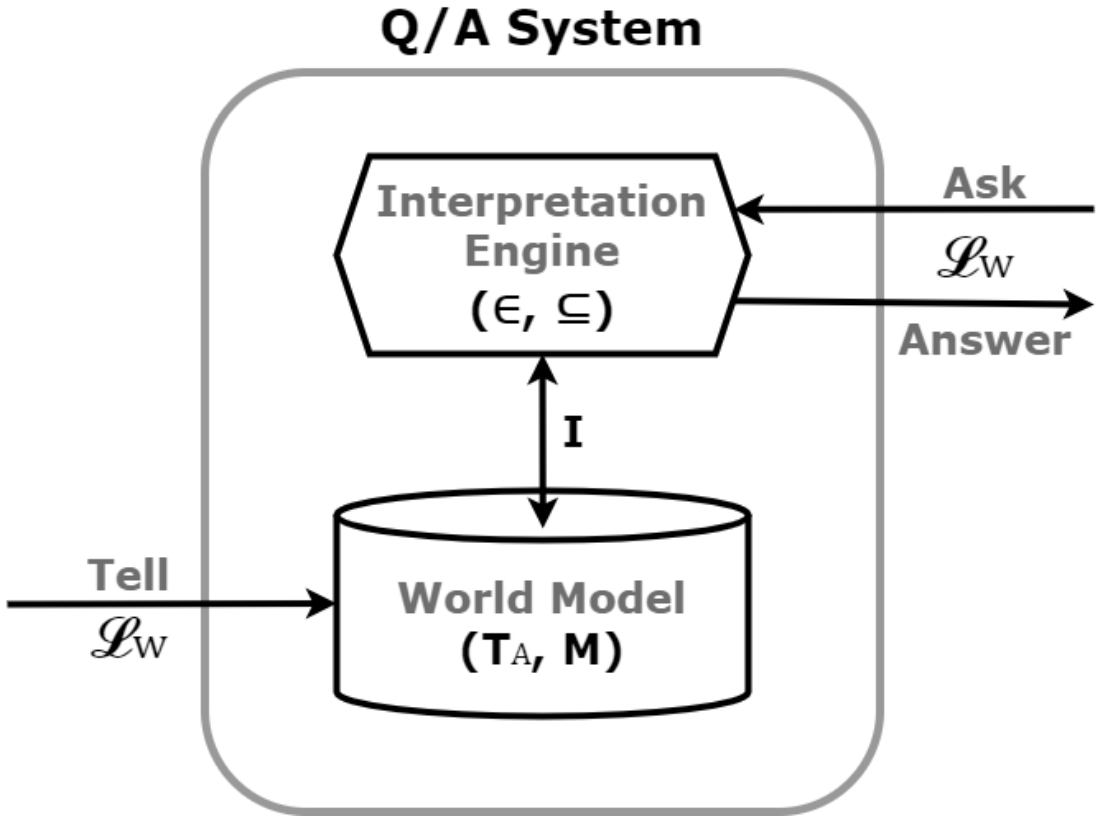
YES!

Intuition (NonMonotonicity) $\Gamma \models w$ and $\Gamma \cup \Sigma$ not$\models w$

NO!

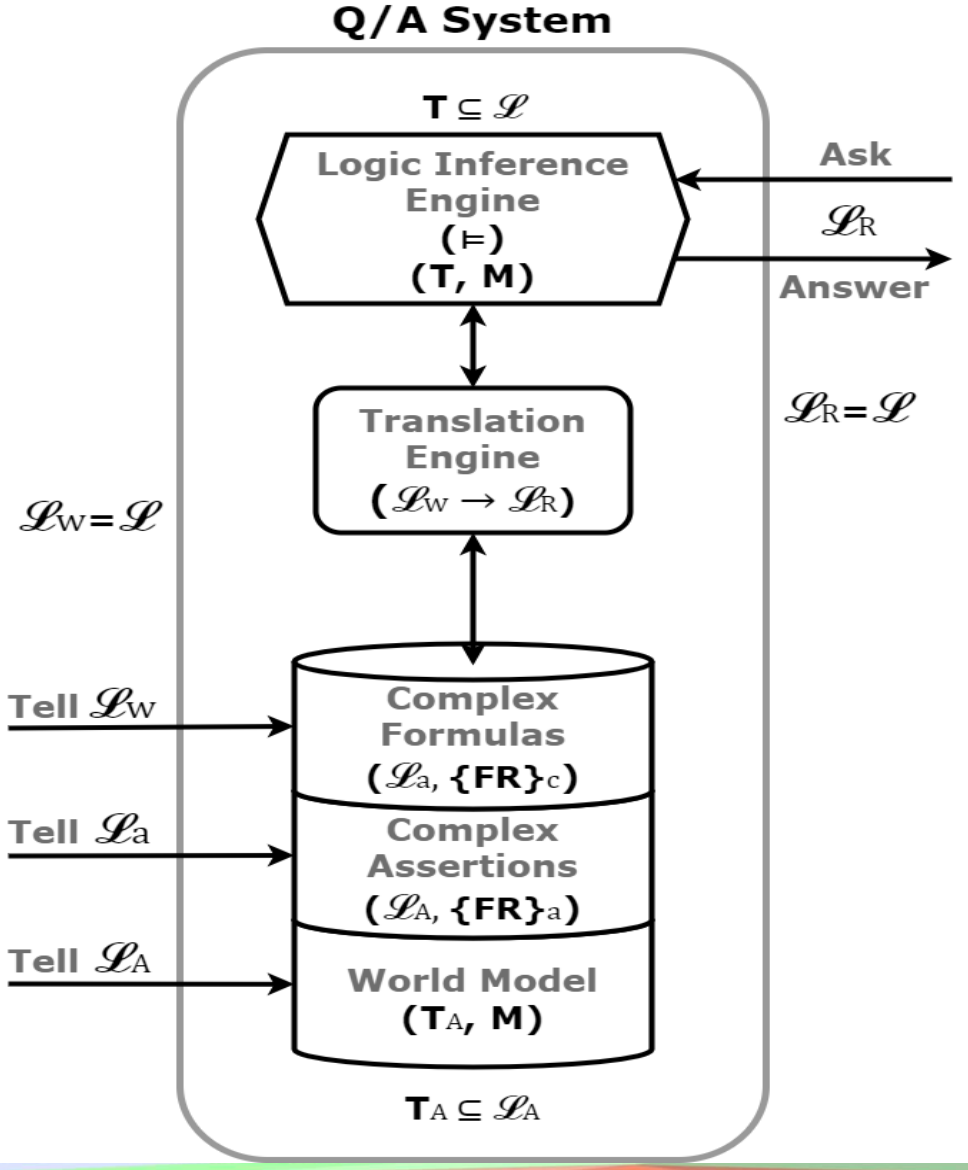
Reasoning as Question Answering (exercises)

Tell Language (basic LOE)
=
Ask Language (basic LOE)



Reasoning as Question Answering (exercises)

Tell Language (RDF)
!=
Ask Language (SPARQL)
(no complex assertions or formulas)



LoE – the Logic of Entities

Sept 13,2023