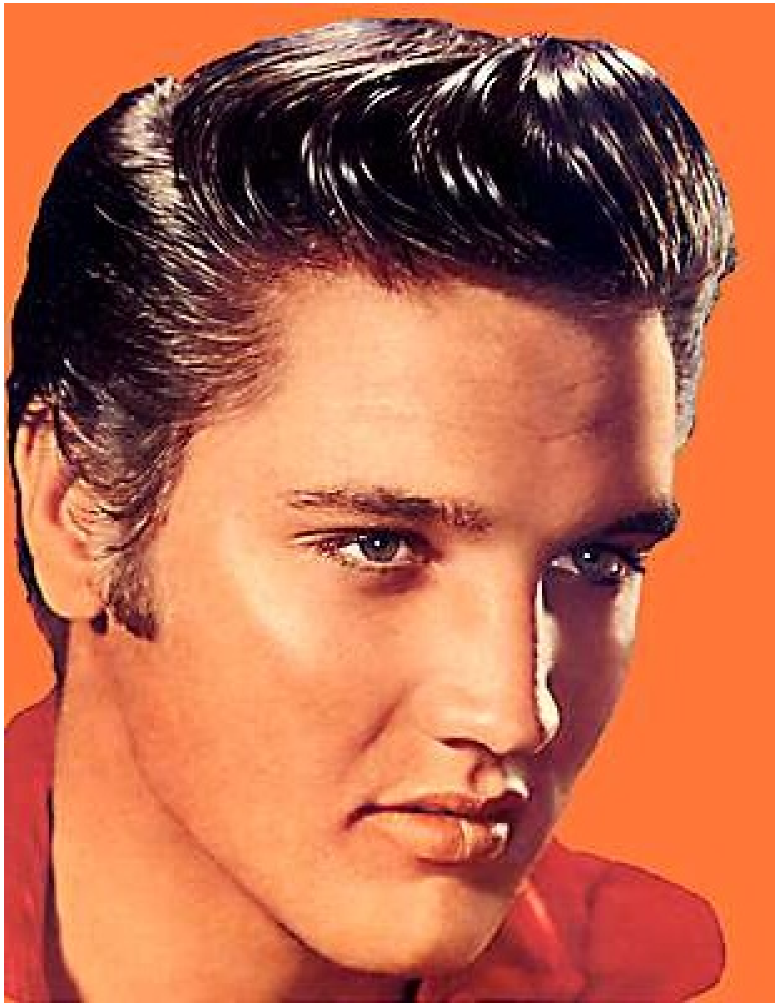
Description Logics

Fabian M. Suchanek

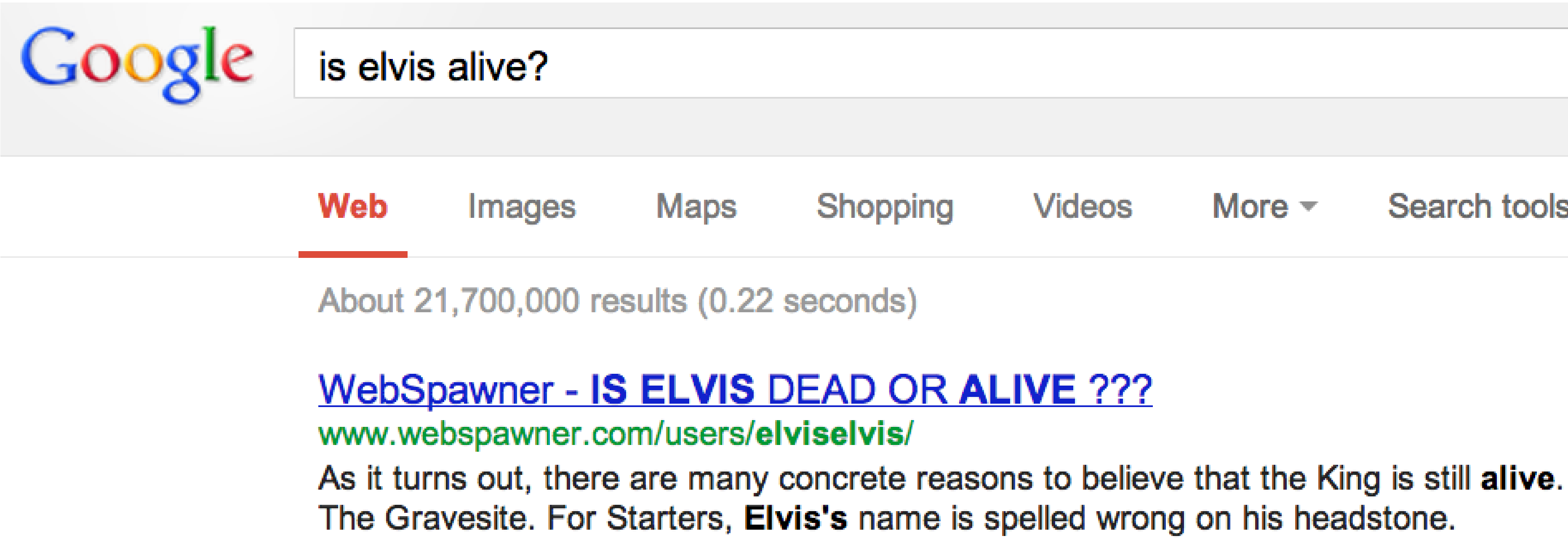
Elvis Presley



Elvis,
when I need you,
I can hear you!

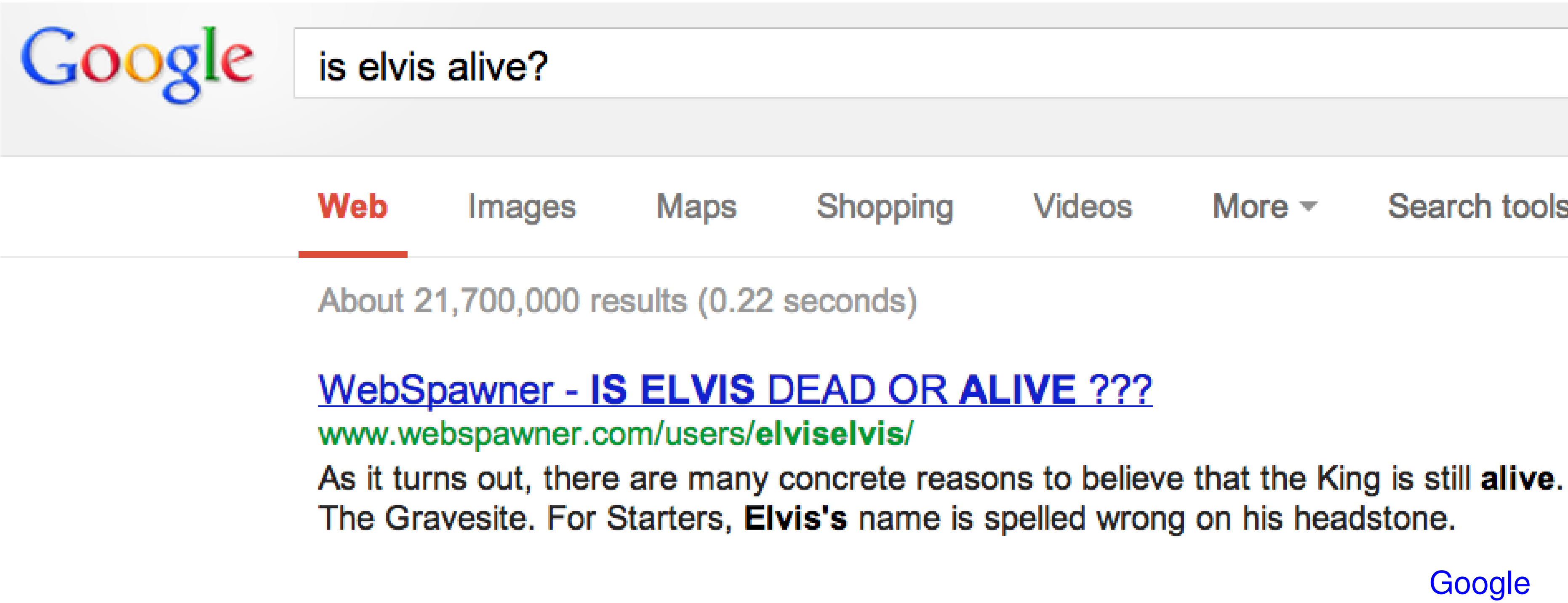
Is Elvis still alive?

Elvis Alive: Google



Google

Elvis Alive: Google



Google: undecided

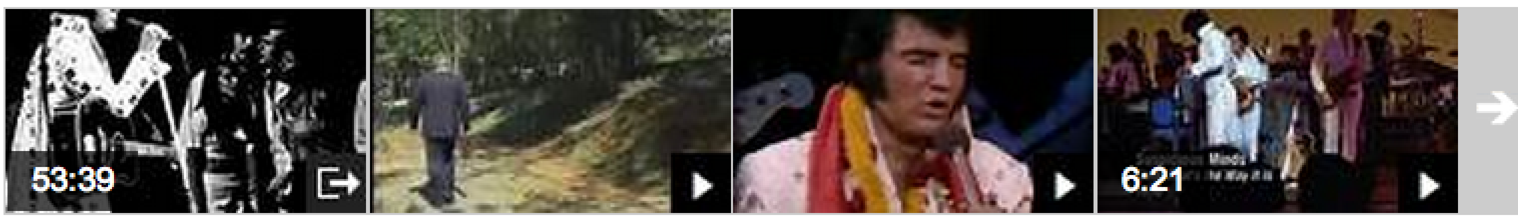
Elvis Alive: Bing

bing

is elvis alive?

23 200 000 ERGEBNISSE Einschränken nach Sprache Einschränken nach Region

[**Videos von is elvis alive?**](#)
bing.com/videos



Elvis Live at Madison Square
YouTube

Elvis Alive
YouTube

Elvis Presley-An American Trilog…
WAt.tv

Elvis Presley Suspicious Mind…
YouTube

elvis is alive! - YouTube Diese Seite übersetzen
www.youtube.com/watch?v=1vMcEJww8os
10 Min. · 139 457 Ansichten · Hinzugefügt am 09/03/2007
We are not interested in fame. Our goal is to show our work and let the world know it: **Elvis is alive**!!!!

Bing

Bing: yes

Elvis Alive: Ask.com

Réponses | Images

is elvis alive?

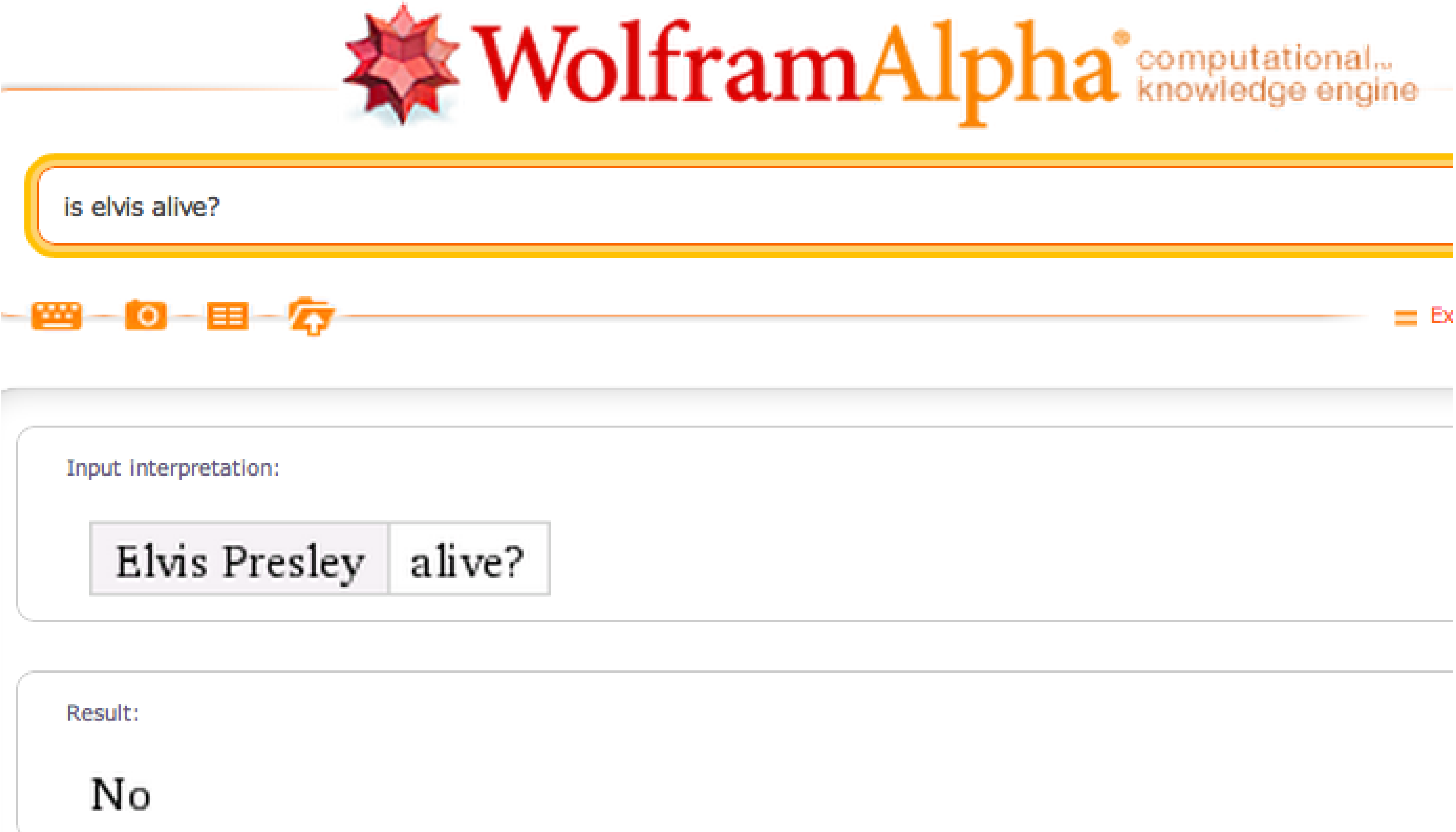
[Proof Elvis is Alive by J. Parra - ...]
www.youtube.com/watch?v=TRBm6q-3SP0
7 Jan 2012 ... In this video, it will show that Elvis is still alive and a doctor who treated him claims that he is still alive.

Ask.com

Ask.com: yes

α

Elvis Alive: Wolfram

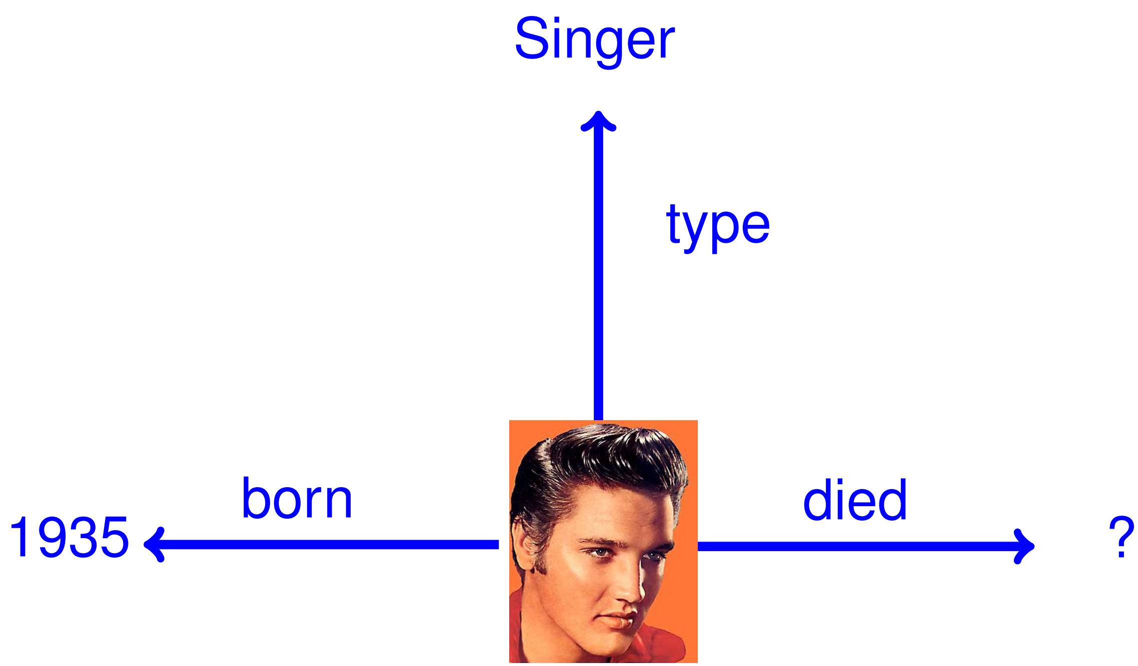


Wolfram Alpha

Wolfram Alpha: no

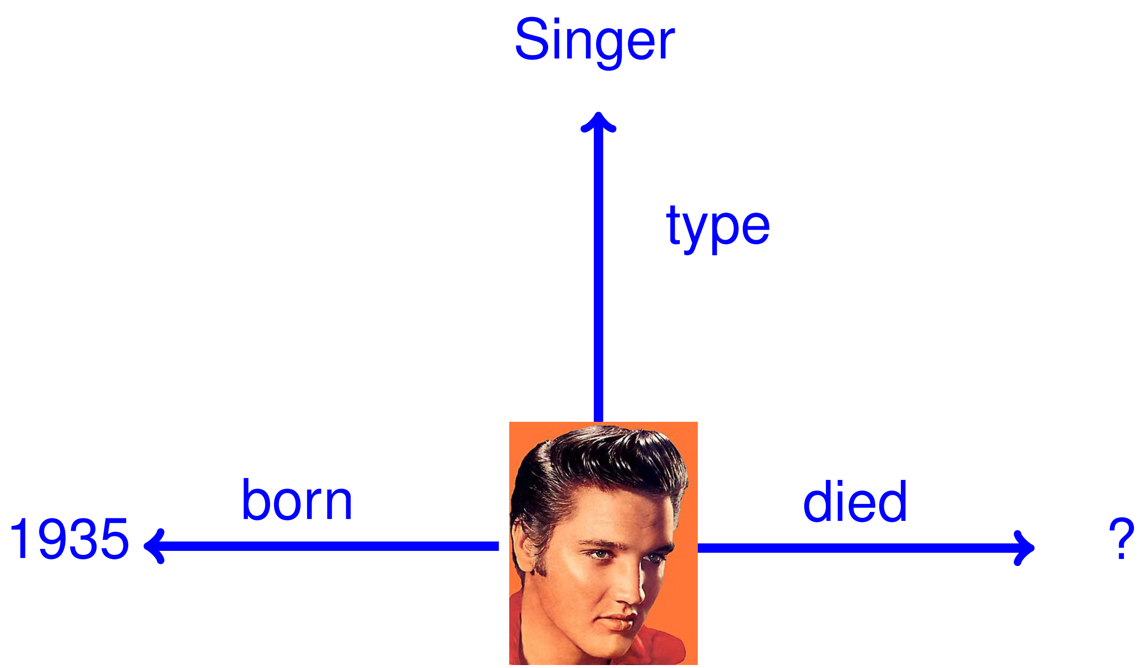
We need structured knowledge

To answer the question, the computer needs structured knowledge: an “ontology”



Def: Ontology

An ontology is a computer-processable collection of knowledge about the world.



Def: Ontology language

An ontology language is a formal languages used to describe and reason on ontologies.

For example: First Order Logic (“FOL”)

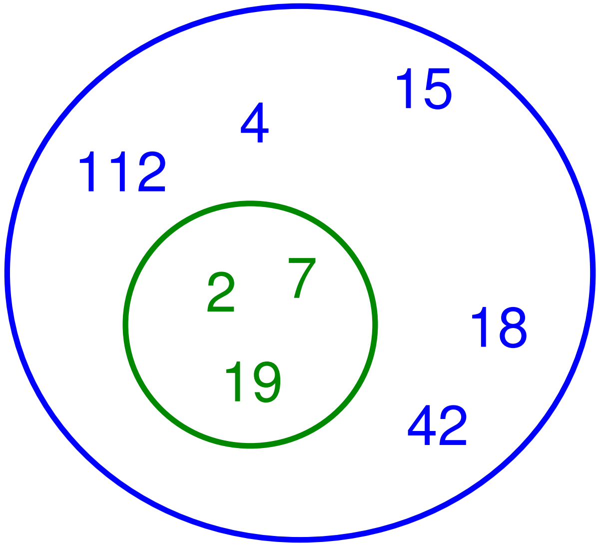
$$\forall x : singer(x) \Rightarrow person(x)$$

Problem: First Order Logic is undecidable!

Def: Decision problem

A decision problem is a question
with a yes-or-no answer. Formally, it is

- a set of inputs (e.g. the natural numbers)
- a subset of the inputs for which the answer shall be "yes" (e.g. the prime numbers)



Def: Undecidable problem

A decision problem is undecidable
if it is impossible to construct a single algorithm
that always terminates and delivers “yes”
on an input iff the input is in the “yes”-subset.

Def: Undecidable problem

A decision problem is undecidable
if it is impossible to construct a single algorithm
that always terminates and delivers "yes"
on an input iff the input is in the "yes"-subset.

Examples:

- Halting problem: Does the program terminate?
- Language equality: Do two context-free grammars generate the same language?
- Hilbert's 10th problem: Does

  $$a_1 x_1^{b_1} + \ldots + a_n x_n^{b_n} = 0$$

  have a solution?

Def: Entscheidungsproblem

The Entscheidungsproblem is the following decision problem:

- input set: set of all FOL formulae
- “yes”-subset: formulae that are tautologies

This decision problem is undecidable.

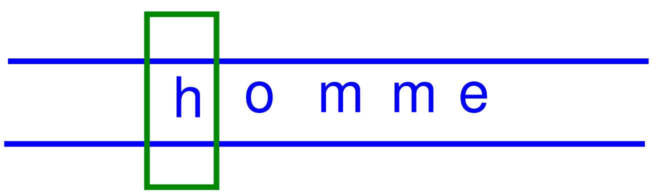
Proof idea: Use a Turing Machine

>Proof

Def: Turing Machine

A Turing Machine is a simplified model of an algorithm (computer program).

Example:



if read ’h’, write ’f’, move right

if read ’o’, write ’e’, move right

if read ’m’ or ’e’, move right

(Plus states; but these could also be encoded in the input)

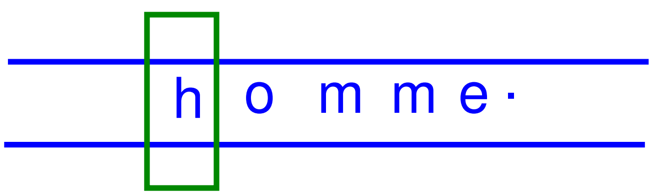
Consensus is that anything that can be computed at all can be computed by a Turing Machine.

(= Church Turing thesis)

Turing Machine in FOL

Every Turing Machine M can be transformed to a FOL formula $\phi(M)$, such that

$$\phi(M) holds \Leftrightarrow M halts$$



M:

if read ’h’, write ’f’, move right

if read ’o’, write ’e’, move right

if read ’m’ or ’e’, move right

$$\phi(M) = \exists t, i : pos(t, i) \land \neg field(t, i, 'h') \land \ldots$$

$$\land(\forall t', i' : pos(t', i') \land field(t', i', 'h') \Rightarrow field(t' + 1, i', 'f'))\ldots$$

Why FOL is undecidable

The Entscheidungsproblem is undecidable:

- input set: set of all FOL formulae
- “yes”-subset: formulae that are tautologies

Proof: Assume that the problem were decidable,
i.e. assume that there is an algorithm A such that

$$A(\phi) = 1 \text{ iff } \phi \text{ is a tautology}$$

Then A can be used to decide whether
$\phi(M)$ is a tautology for a Turing Machine M.
=$>$ A decides the halting problem (impossible)

FOL is semi-decidable

We can construct an algorithm that enumerates all correct proofs:

$$\forall x : p(x) \Rightarrow p(x)$$

$$\forall x : p(x) \Rightarrow p(x) \vee q(x)$$

$$\forall x : p(x) \wedge q(x) \Rightarrow p(x) \vee q(x)$$

...

=> If the algorithm terminates, it found the input formula, and the input formula is a tautology

=> If the algorithm does not terminate, the formula is either not a tautology or we need to wait longer.

"Semi-decidability"

Decidable subsets of FOL

The Entscheidungsproblem is decidable

if we consider only certain subsets of FOL.

Examples:

- Propositional logic (i.e., no quantifiers)
- Horn rules
- Bernays-Schönfinkel formulae,

  i.e., formulae of the form

  $$\exists * \forall * \phi$$

  without function symbols
- Description logic

Def: Description logic

A description logic (DL) is a special logic,

which usually

- is a subset of FOL
- uses only unary and binary predicates (relations)
- is decidable
- comes with a special syntax

  that does not use variables

Example:

DL formula

$$human \sqsubseteq man \sqcup woman$$

$$\forall x : human(x) \Rightarrow man(x) \vee woman(x)$$

Equivalent FOL formula

>SHIQ

Description logics

There are several description logics.

They allow or disallow, for example,

- union of concepts (“U”)
- functional predicates (“F”)
- complex concept negation (“C”)
- cardinality restrictions (“N”)

This yields abbreviations for these logics, such as $SHIQ$.

We will look at $SHOIQ^{(D)}$, which is the basis of OWL 2, the language used on the Semantic Web.

Description logic syntax

Different syntaxes have been developed
to say the same thing in OWL:

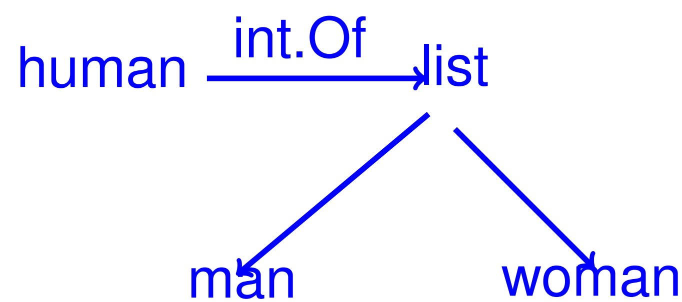
$human \equiv man \sqcup woman$

We will stick
to this one

ObjectIntersectionOf(man woman)

<human> owl:intersectionOf <list>
<list> rdf:_1 <man>
<list> rdf:_2 <man>

Class: Human
EquivalentTo: Man and Woman



Intersection

DL is primarily concerned with describing sets of entities:

$X \sqcap Y$ the class of things that are both X and Y

$$father \equiv parent \sqcap malePerson$$

Intersection

DL is primarily concerned with describing sets of entities:

$X \sqcap Y$ the class of things that are both X and Y

$$father \equiv parent \sqcap malePerson$$

This corresponds to the First-Order Formula

$$\forall x : father(x) \Leftrightarrow parent(x) \land malePerson(x)$$

Intersection

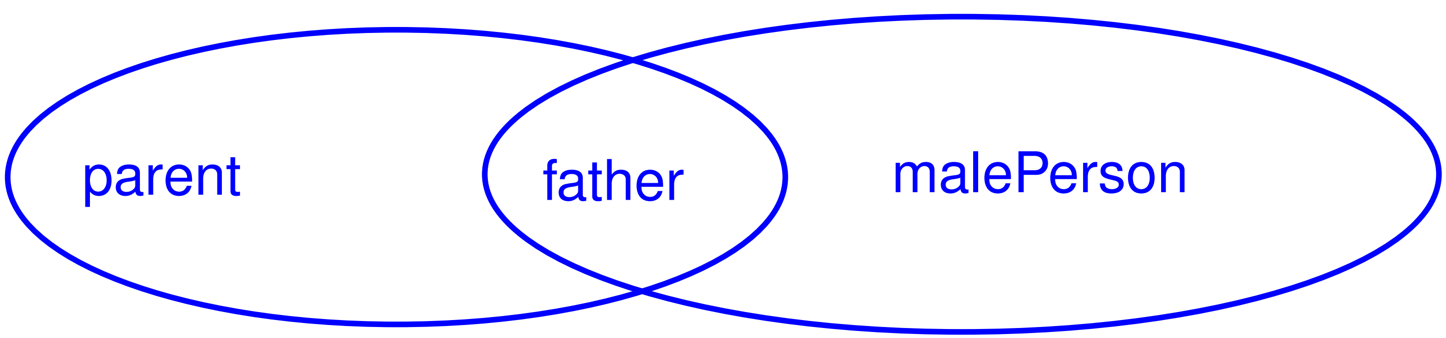
DL is primarily concerned with describing sets of entities:

$X \sqcap Y$ the class of things that are both X and Y

$$father \equiv parent \sqcap malePerson$$

This corresponds to the First-Order Formula

$$\forall x : father(x) \Leftrightarrow parent(x) \wedge malePerson(x)$$



Intersection, Union, Compl. & Task

$X \sqcap Y$	the class of things that are both X and Y
$X \sqcup Y$	the class of things that are X or Y
$\neg X$	the class of things that are not X

>task

Intersection, Union, Compl. & Task

$X \sqcap Y$ the class of things that are both X and Y

$X \sqcup Y$ the class of things that are X or Y

$\neg X$ the class of things that are not X

Task: Given these classes

person, parent, hardRockSinger, softRockSinger, happyPerson, marriedPerson, malePerson

Define

rocksinger, unmarried-rock-singing-father, non-rock-singing-person

Universal Restriction

$\forall R.C$ the class of things where all outgoing R-links lead to a C

a relation a class

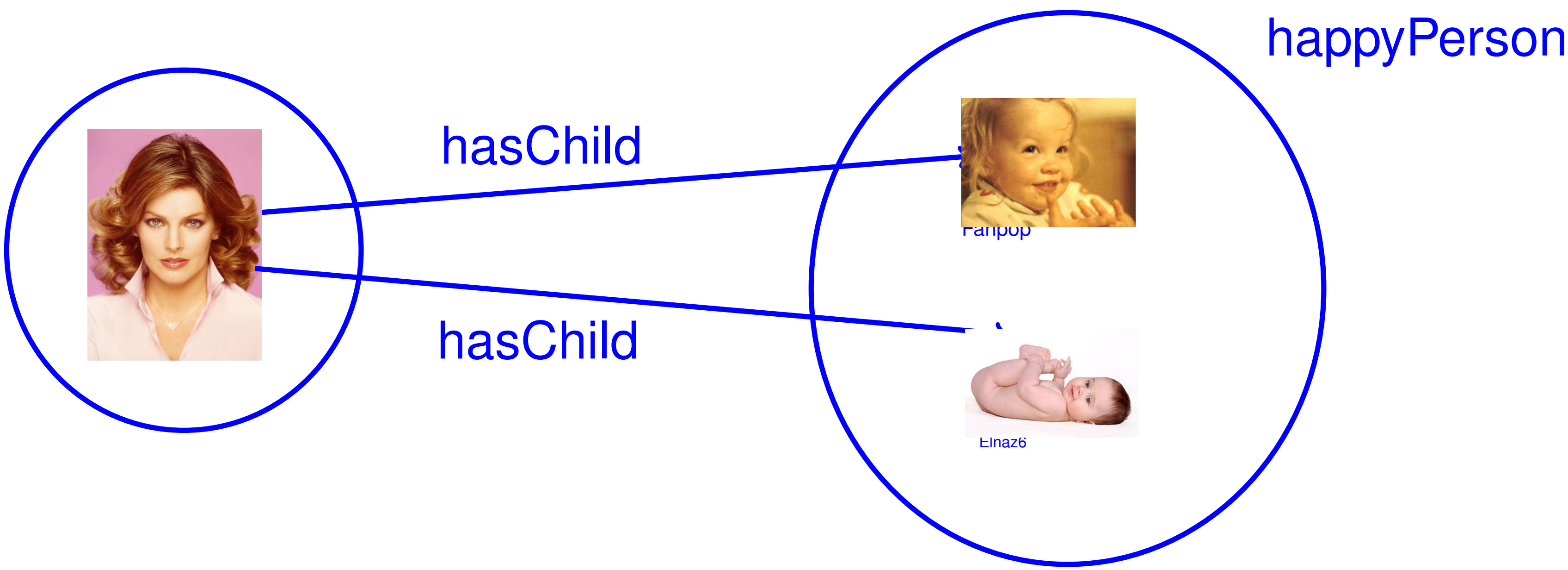
$hasOnlyHappyKids \equiv \forall hasChild.HappyPerson$

Universal Restriction

$\forall R.C$ — the class of things where all outgoing R-links lead to a C

a relation — a class

$$hasOnlyHappyKids \equiv \forall hasChild.HappyPerson$$



Restriction

$\forall R.C$ the class of things where all outgoing R-links lead to a C

a relation a class

$$hasOnlyHappyKids \equiv \forall hasChild.HappyPerson$$

This corresponds to

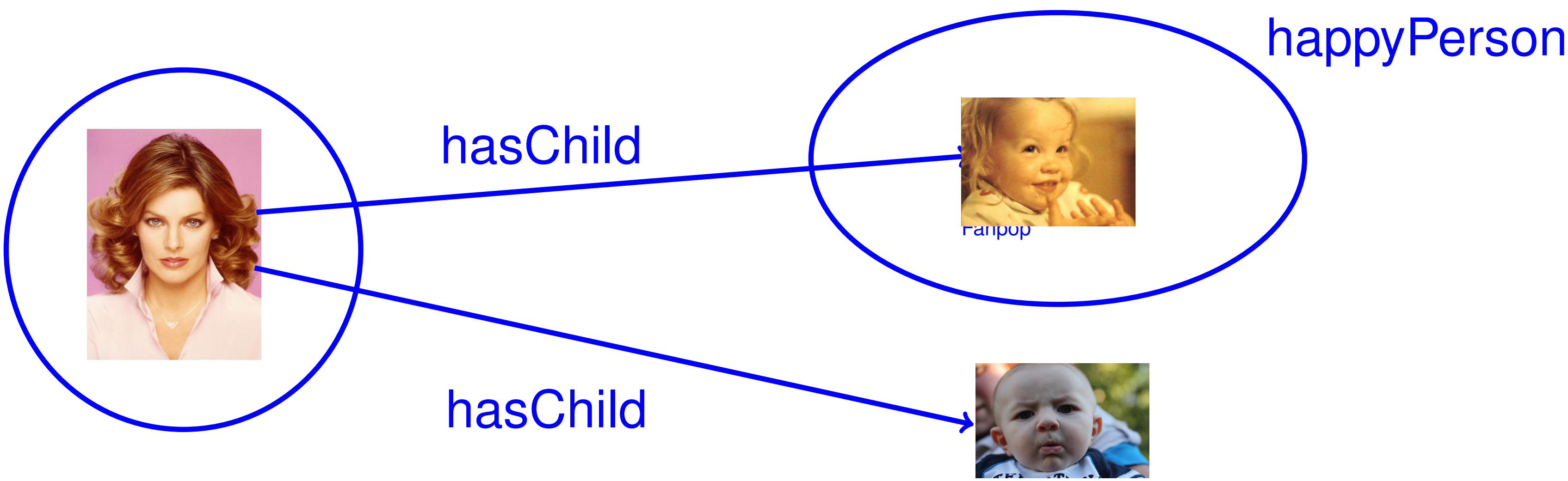
$$\forall x : hasOnlyHappyKids(x)$$

$$\Leftrightarrow \forall y : hasChild(x, y) \Rightarrow happyPerson(y)$$

Existential Restriction

$\exists R.C$ the class of things where there exists an outgoing R-link leading to a C

$$hasHappyKid \equiv \exists hasChild.HappyPerson$$



Existential Restriction

$\exists R.C$ the class of things where there exists an outgoing R-link leading to a C

$$hasHappyKid \equiv \exists hasChild.HappyPerson$$

This corresponds to

$$\forall x : hasHappyKid(x)$$

$$\Leftrightarrow \exists y : hasChild(x, y) \wedge happyPerson(y)$$

>task
>end

Task: Restrictions

$\exists R.C$	the class of things where there exists an outgoing R-link leading to a C
$\forall R.C$	the class of things where all outgoing R-links lead to a C

Given:

singer, happyPerson, hasChild

Build:

- the class of singers with only happy children
- the class of happy people who have at least one happy singer as child

$>$end

Concept Inclusions

$X \sqsubseteq Y$ every X is a Y

$$singer \sqsubseteq person$$

Concept Inclusions

$X \sqsubseteq Y$ every X is a Y

$$singer \sqsubseteq person$$

$$rocksinger \sqsubseteq singer \sqcap rockFan$$

This corresponds to

$$\forall x : rocksinger(x) \Rightarrow singer(x) \land rockFan(x)$$

Concept Inclusions

$X \sqsubseteq Y$ every X is a Y

$$singer \sqsubseteq person$$

$$rocksinger \sqsubseteq singer \sqcap rockFan$$

This corresponds to

$$\forall x : rocksinger(x) \Rightarrow singer(x) \wedge rockFan(x)$$

$$happySinger \sqsubseteq singer \sqcap \forall hasChild.Happy$$

>end

Concept and Role Assertions

$R(x, y)$ x and y stand in relationship R

$$hasChild(elvis, lisa)$$

$C(x)$ x is an instance of class C

$$Singer(elvis)$$

$$HappySinger \equiv Singer \sqcap \forall hasChild.Happy$$

$$HappySinger(elvis)$$

>task
>end

Task: Description Logic

Class constructors:

$$X \sqcap Y \qquad X \sqcup Y \qquad \neg X \qquad \forall R.C \qquad \exists R.C$$

Assertions:

$$X \equiv Y \qquad X \sqsubseteq Y \qquad C(x) \qquad R(x,y)$$

Given: male, person, happyPerson
marriedTo, hasChild

- build the class of married people
- build the class of people married to at least 1 happy person
- build the class of happy male married people
- say that married people are happy

>end

Cardinality restrictions

$\geq nR.C$ the class of those things that have at least n outgoing R links to a C

$\leq nR.C$ the class of those things that have at most n outgoing R links to a C

Singers with more than 10 happy children

$$singer \sqcap \geq 10 hasChild.Happy$$

Singers with exactly 10 happy children:

>end

Enumerations

$\{a, b, c\}$ the class of a, b, and c

$\exists marriedTo.\{priscilla\}$

(the class of Priscilla's husbands)

>end

Top and Bottom & Task

$\top$ the class of all things ("top")

$\bot$ the empty class ("bottom")

Everybody has only happy children:

$$\top \sqsubseteq \forall hasChild.Happy$$

Nobody has only happy children:

$$\forall hasChild.Happy \sqsubseteq \bot$$

Task: Express

People with an unhappy child are not happy.

People whose children are all male are not happy

>end

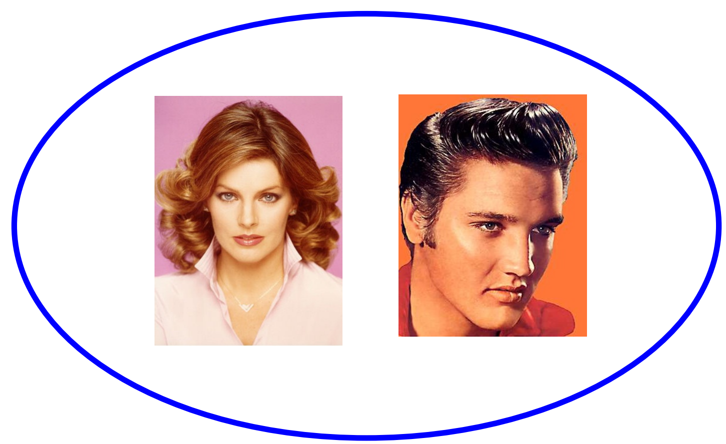
What DL can say

DL basically reasons about properties such as

- having a link to an element of a class

$\exists hasChild.Female$

“Those that have
a female child”



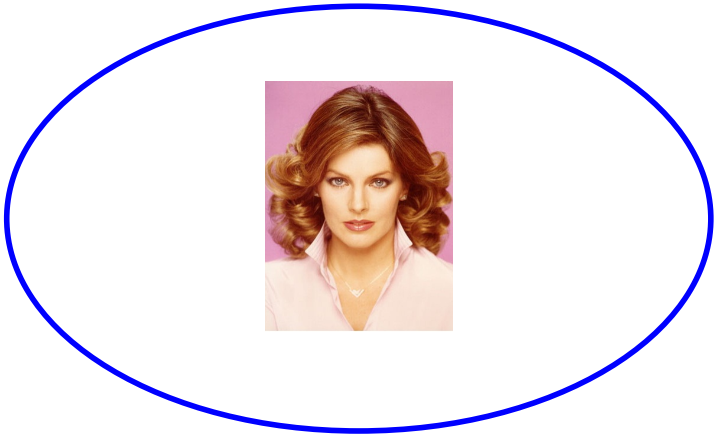
What DL can say

DL basically reasons about properties such as

- having a/all links to an element of a class
- having at most or at least n links of a certain type

$\geq 2 hasChild$

"Those who have more than 1 child"

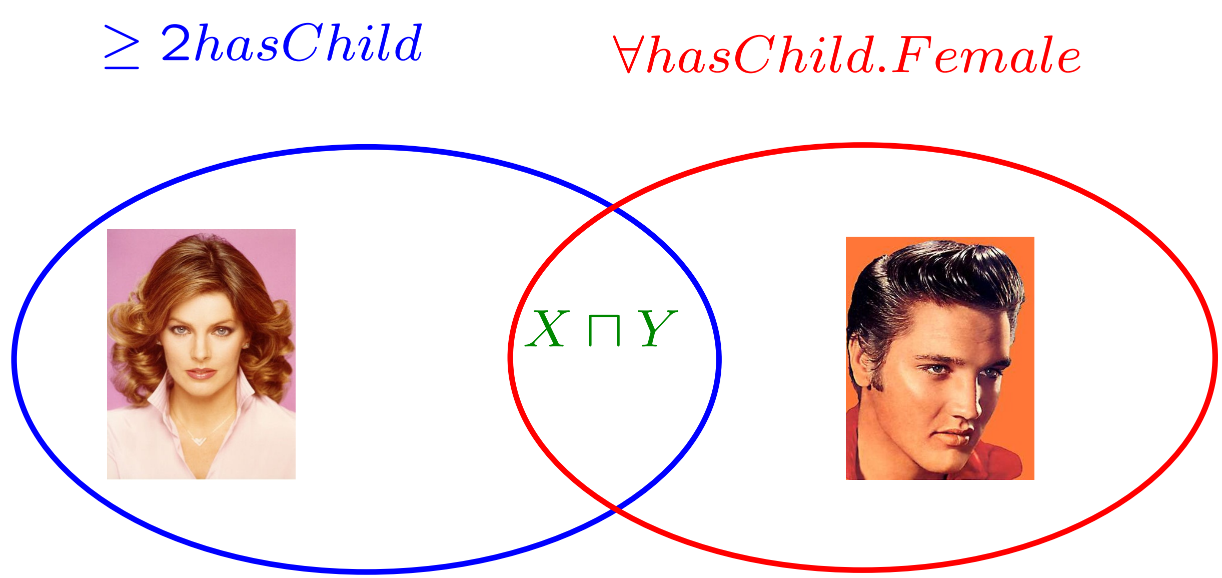


What DL can say

DL basically reasons about properties such as

- having a/all links to an element of a class
- having at most or at least n links of a certain type
- being in two classes at the same time

"Those that have more than 1 child and have only female children"



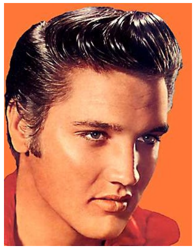
What DL can say

DL basically reasons about properties such as

- having a/all links to an element of a class
- having at most or at least n links of a certain type
- being in two classes at the same time
- belonging to such a class

$\forall hasChild.Female$

"Elvis has only female children"



$elvis : \forall hasChild.Female$

What DL can say

DL basically reasons about properties such as

- having a/all links to an element of a class
- having at most or at least n links of a certain type
- being in two classes at the same time
- belonging to such a class
- standing in a relationship

All of these statements correspond to
First Order Logic formulae.
The Entscheidungsproblem is decidable
on this subset of FOL!

>end

Reasoning tasks

Classical reasoning tasks in DL are:

- Is the knowledge base consistent?

$$singer \sqsubseteq person$$

$$dog \sqsubseteq \neg person$$

$$dog(bello)$$

Reasoning tasks

Classical reasoning tasks in DL are:

- Is the knowledge base consistent?

$$singer \sqsubseteq person$$

$$dog \sqsubseteq \neg person$$

$$dog(bello)$$

$$person(bello)$$

Reasoning tasks

Classical reasoning tasks in DL are:

- Is the knowledge base consistent?
- Is one class a subclass of another class?

$$\exists hasChild.\{lisa\} \sqsubseteq singer?$$

(Are all parents of Lisa singers?)

Reasoning tasks

Classical reasoning tasks in DL are:

- Is the knowledge base consistent?
- Is one class a subclass of another class?

$$\exists hasChild.\{lisa\} \sqsubseteq singer?$$

(Are all parents of Lisa singers?)

Can be reduced to the consistency problem:

$$leftClass \equiv \exists hasChild.\{Lisa\}$$

$$notRightClass \equiv \neg singer$$

$$leftClass(bob)$$

$$notRightClass(bob)$$

Reasoning tasks

Classical reasoning tasks in DL are:

- Is the knowledge base consistent?
- Is one class a subclass of another class?

$$\exists hasChild.\{lisa\} \sqsubseteq singer?$$

(Are all parents of Lisa singers?)

Can be reduced to the consistency problem:

$$leftClass \equiv \exists hasChild.\{Lisa\}$$

$$notRightClass \equiv \neg singer$$

$$leftClass(bob)$$

$$notRightClass(bob)$$

More precisely, we ask: Does it follow necessarily from the KB that one class is a subclass of the other?
If so, we get a contradiction here.

Reasoning tasks

Classical reasoning tasks in DL are:

- Is the knowledge base consistent?
- Is one class a subclass of another class?
- Is an individual an instance of a class?

$singer(elvis)?$ (Is Elvis a singer?)

Reasoning tasks

Classical reasoning tasks in DL are:

- Is the knowledge base consistent?
- Is one class a subclass of another class?
- Is an individual an instance of a class?

$singer(elvis)?$ (Is Elvis a singer?)

Can be reduced to the consistency problem:

$notSinger \equiv \neg singer$

$notSinger(elvis)$

Reasoning tasks

Classical reasoning tasks in DL are:

- Is the knowledge base consistent?
- Is one class a subclass of another class?
- Is an individual an instance of a class?

$singer(elvis)?$ (Is Elvis a singer?)

Can be reduced to the consistency problem:

$notSinger \equiv \neg singer$

$notSinger(elvis)$

More precisely, we ask: Does it follow necessarily from the KB that Elvis is a singer? If so, this assertion will cause a contradiction.

Reasoning tasks

Classical reasoning tasks in DL are:

- Is the knowledge base consistent?
- Is one class a subclass of another class?
- Is an individual an instance of a class?
- Does an individual have a certain property?

$$elvis : singer \sqcap \exists sings.GoodSong?$$

Reasoning tasks

Classical reasoning tasks in DL are:

- Is the knowledge base consistent?
- Is one class a subclass of another class?
- Is an individual an instance of a class?
- Does an individual have a certain property?

$$elvis : singer \sqcap \exists sings.GoodSong?$$

Can be reduced to the consistency problem:

$$notGoodSinger \equiv \neg(singer \sqcap \exists sings.GoodSong)$$

$$notGoodSinger(elvis)$$

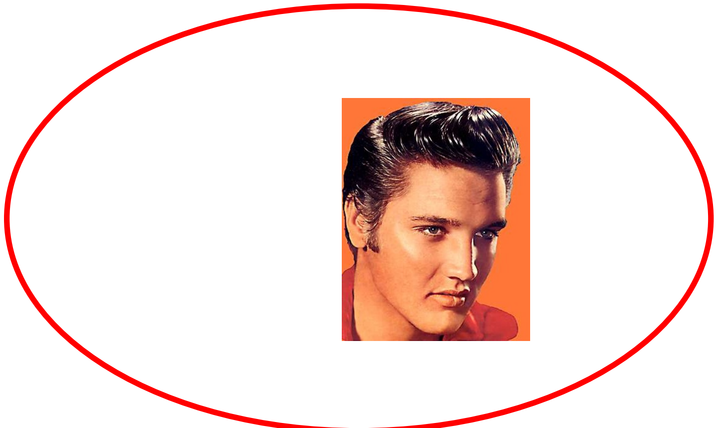
DL Summary

OWL DL is a decidable subset of First Order Logic. It allows describing properties of objects in a manner inspired by set theory.

There are a number of free OWL DL reasoners available online:

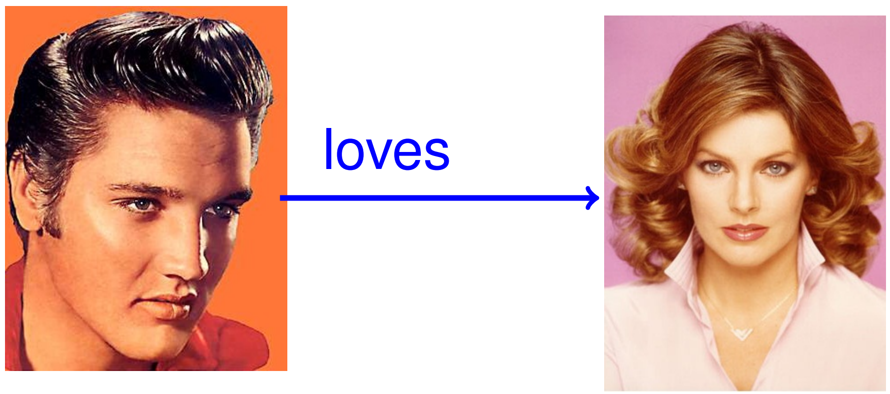
- Pellet
- FaCT++
- Prova

$\forall hasChild.Female$



Course Summary

- An ontology is a formal collection of knowledge about the world



- First-Order-Logic is undecidable
- Description Logics are decidable fragments of First Order Logic