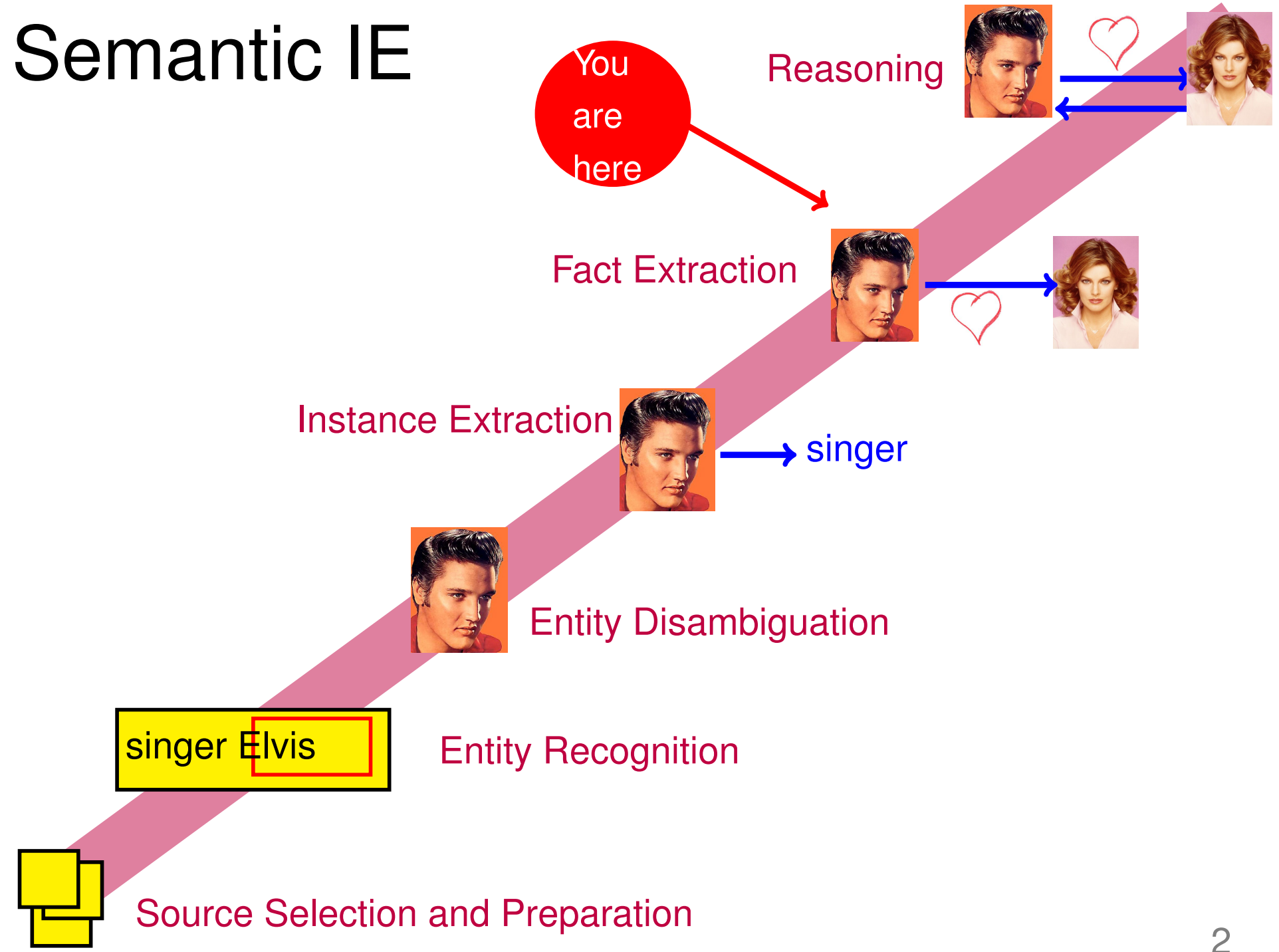
Information Extraction by Reasoning

Fabian M. Suchanek

Semantic IE

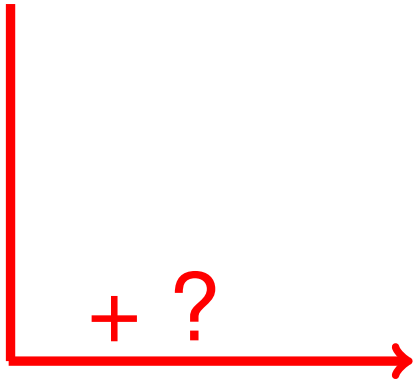

You are here
Reasoning
Fact Extraction
Instance Extraction
singer
Entity Disambiguation
singer Elvis
Entity Recognition
Source Selection and Preparation

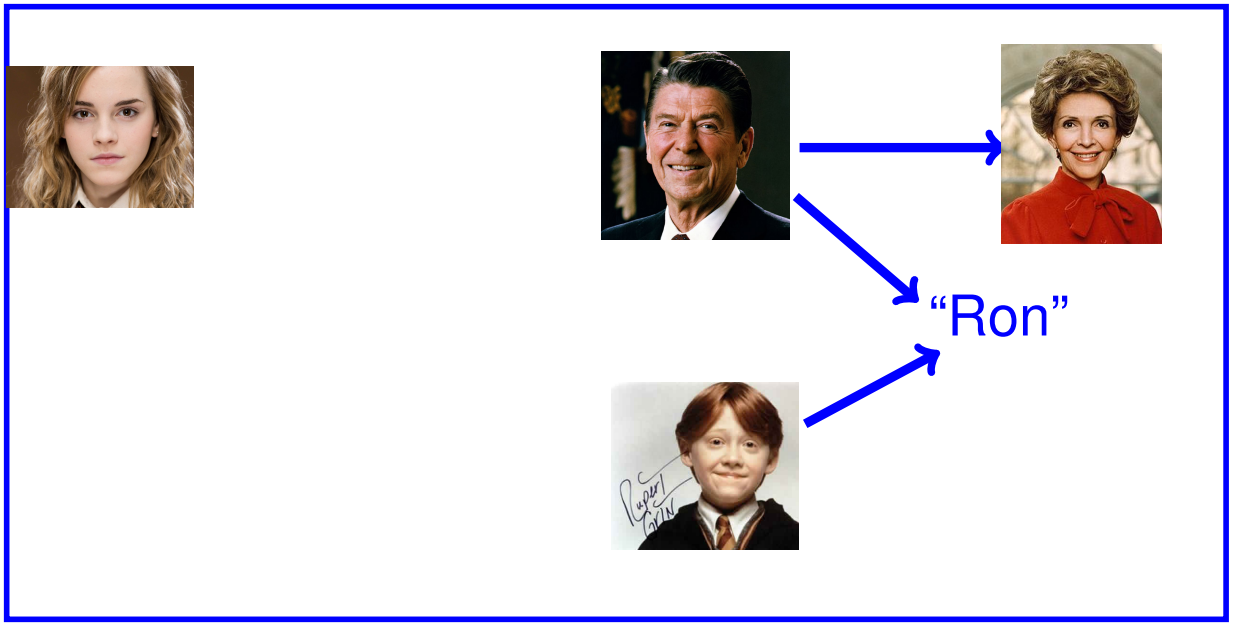
Problems in extending a KB

"Hermione is married to Ron"

Problems in extending a KB

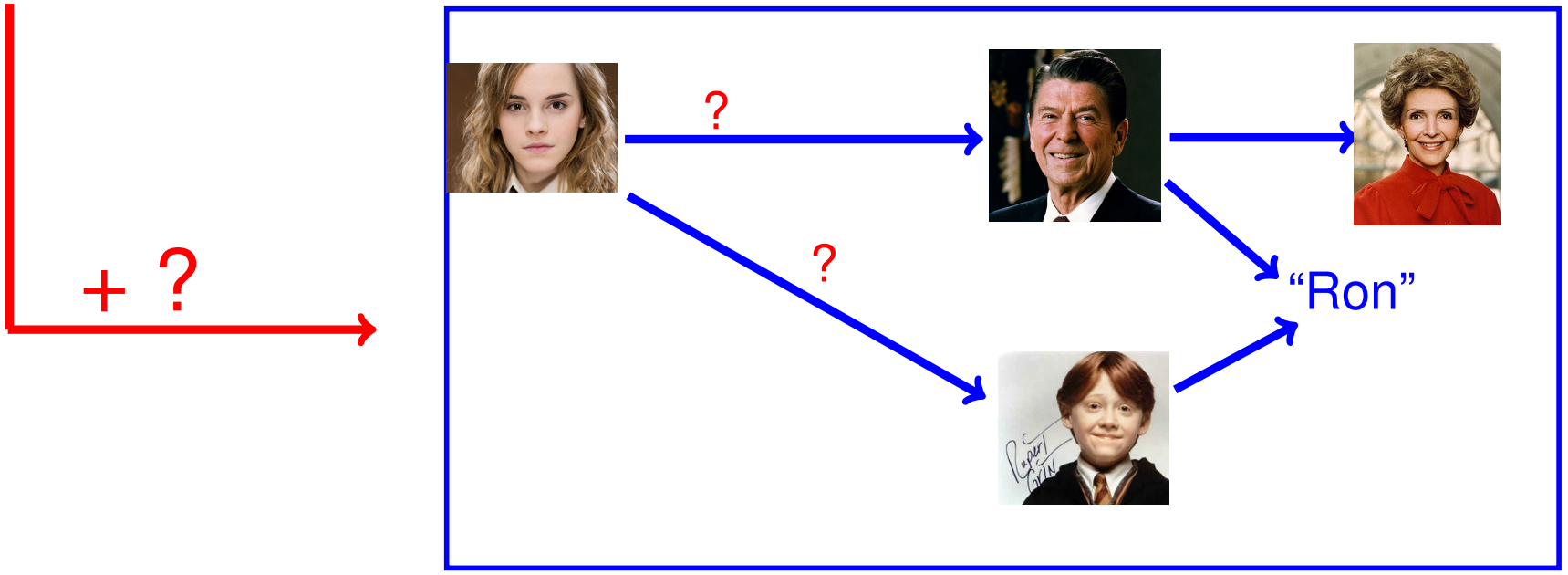
"Hermione is married to Ron"





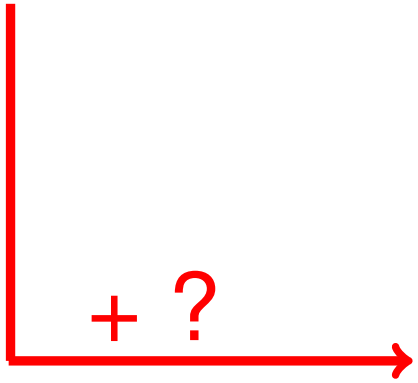
Problems in extending a KB

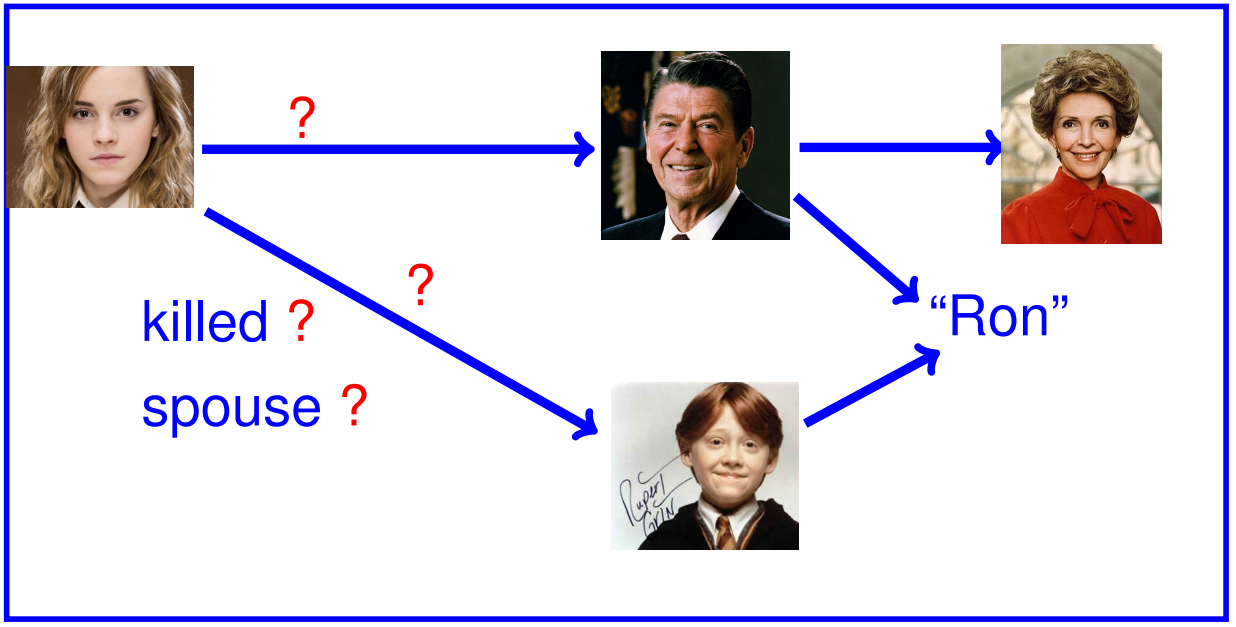
"Hermione is married to Ron"



Problems in extending a KB

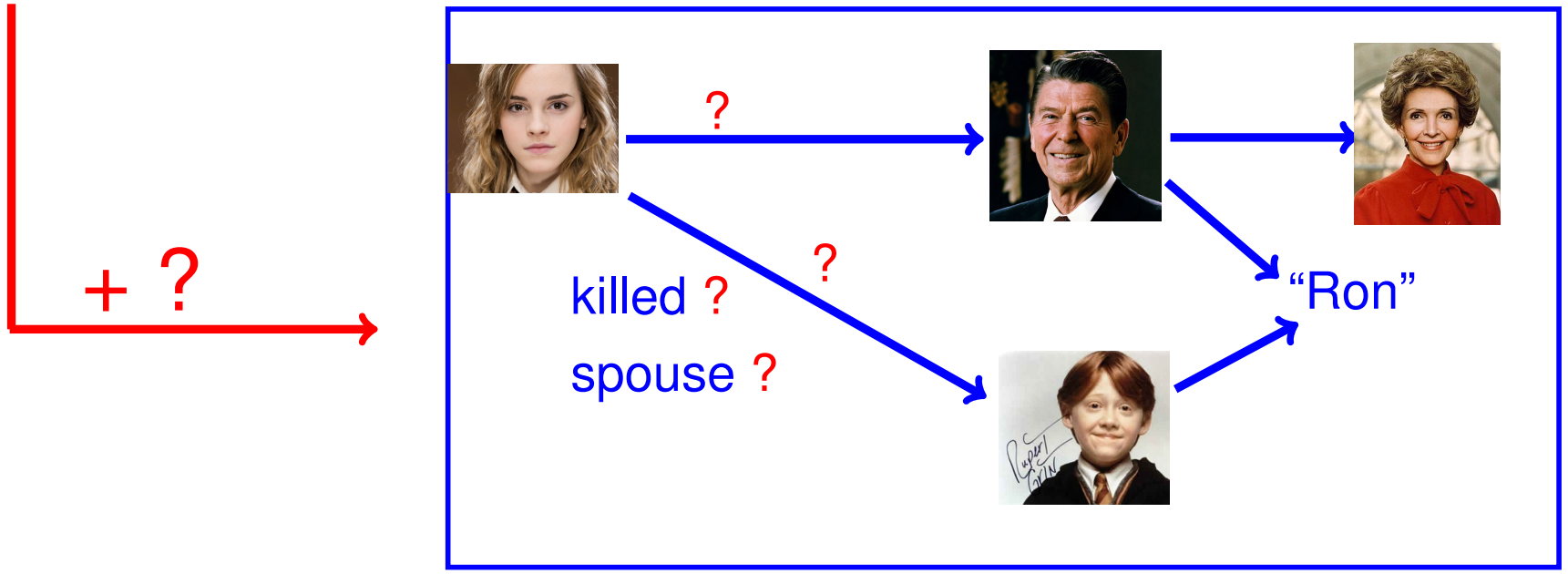
"Hermione is married to Ron"





Problems in extending a KB

"Hermione is married to Ron"

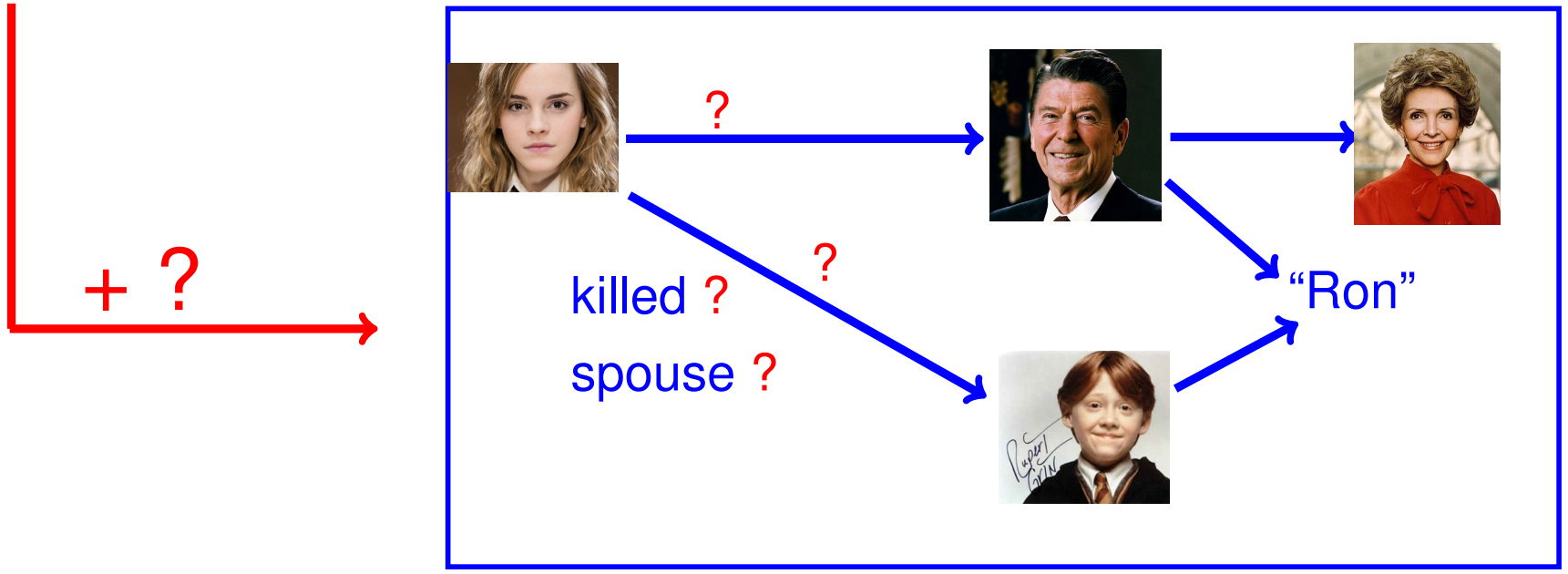


IE faces at least 3 problems:

- Understand patterns ("X is married to Y" = killed(X,Y)?)
- Disambiguate entities ("Ron"= Ronald Reagan?)
- Resolve inconsistencies (Reagan married to 2 women?)

Problems in extending a KB

"Hermione is married to Ron"



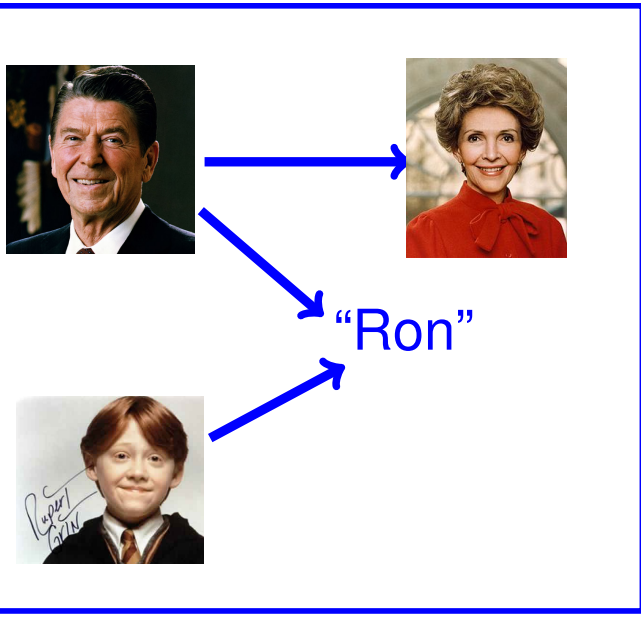
- Disambiguation avoids inconsistency
- Pattern helps disambiguation
- Consistency helps finding pattern

=> Solve all 3 problems together!

Idea: Solve all problems together

"Hermione is married to Ron"

transform everything to logical formulas



$A \wedge B \Rightarrow C$

Find best conclusion

$hasSpouse(Hermione, RonWeasley)$

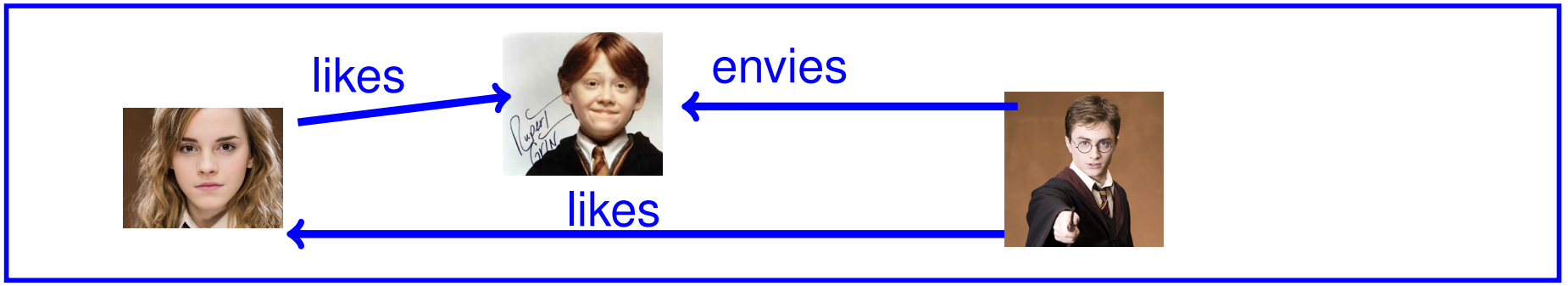
Implications>13

Refresh: Atoms and KBs

An atom A is a propositional statement. A is a positive literal, and $\neg A$ is a negative literal. The polarity is positive for A, and negative for $\neg A$.

A positive literal holds ("is true") in a KB, if it appears in the KB.

A negative literal $\neg A$ holds in a KB if A does not hold.



$likes(Hermione, Ron)$?

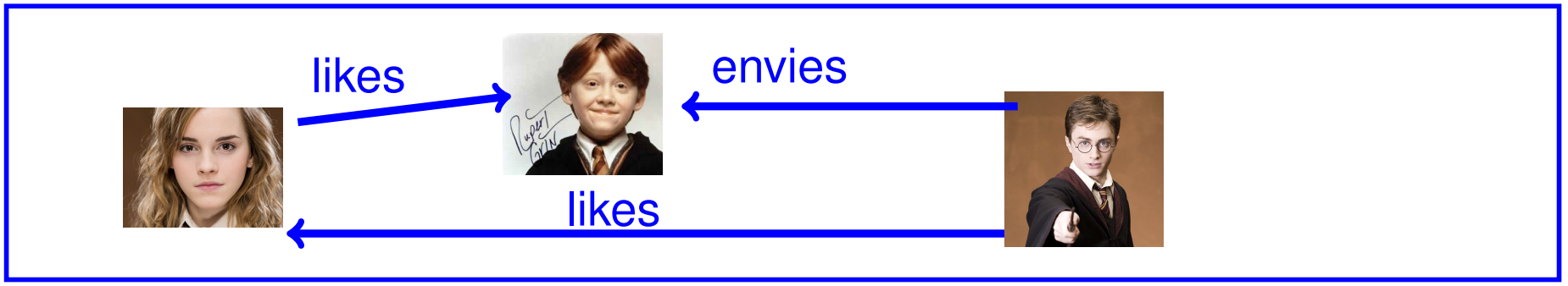
$\neg envies(Ron, Harry)$?

Refresh: Atoms and KBs

An positive literal holds ("is true") in a KB, if it appears in the KB.

A negative literal $\neg A$ holds in a KB if A does not hold.

A conjunction $A \wedge B \wedge ... \wedge Z$ holds in a KB, if all of its elements hold.



$likes(Hermione, Ron)$?

$\neg envies(Ron, Harry)$?

$\neg envies(Ron, Harry) \wedge likes(Harry, Hermione)$ T

$envies(Harry, Ron) \wedge likes(Hermione, Ron) \wedge likes(Harry, Elvis)$ F

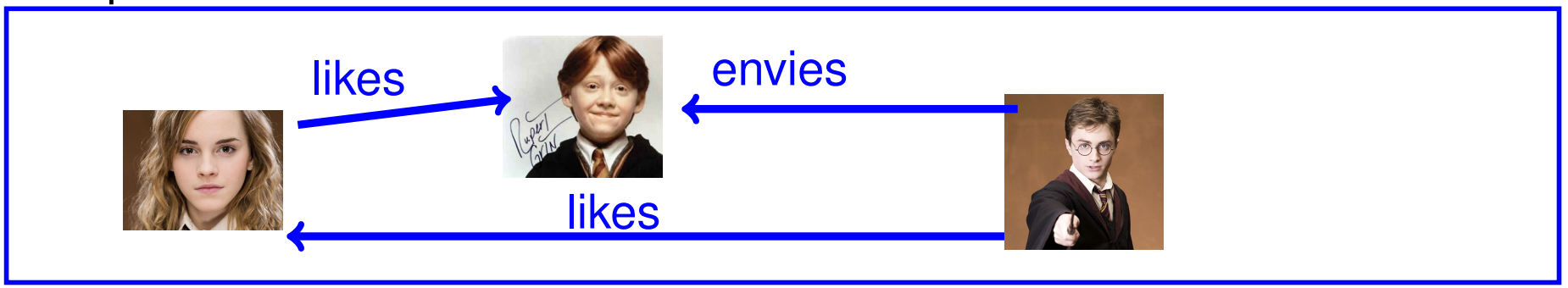
Refresh: Implications

An positive literal holds ("is true") in a KB, if it appears in the KB.

A negative literal $\neg A$ holds in a KB if A does not hold.

A conjunction $A \wedge B \wedge ... \wedge Z$ holds in a KB, if all of its elements hold.

An implication $\vec{B} \Rightarrow H$ holds in a KB if $\vec{B}$ does not hold or H holds.



$likes(Hermione, Ron) \Rightarrow likes(Harry, Ron)$ (T ⇒ F = F)

$likes(Harry, Ron) \Rightarrow hasSpouse(Harry, Hermione)$

$\neg likes(Hermione, Harry) \Rightarrow envies(Harry, Ron)$

$\neg likes(Hermione, Harry) \Rightarrow hasSpouse(Harry, Ron)$

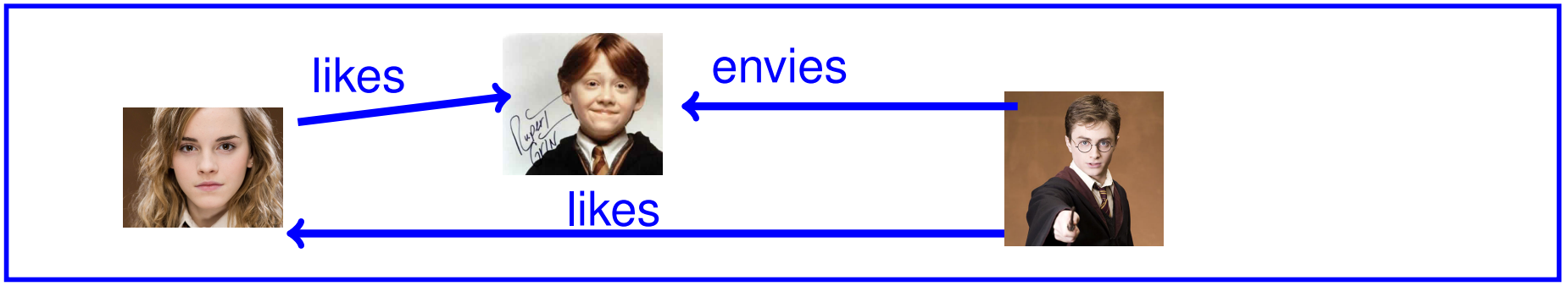
$likes(Herm., Ron) \wedge \neg envies(Harry, Ron) \Rightarrow ffe(Harry, Ron)$

Def: Rules, Disjunctions, Clauses

An implication (also: rule) $B_1 \wedge ... \wedge B_n \Rightarrow H$

is equivalent to a disjunction $\neg B_1 \vee ... \vee \neg B_n \vee H$

which we also write as a clause $\{\neg B_1, ..., \neg B_n, H\}$.



$likes(Hermione, Ron) \Rightarrow likes(Harry, Ron)$

is equivalent to

$\neg likes(Hermione, Ron) \vee likes(Harry, Ron)$

is equivalent to

$\{\neg likes(Hermione, Ron), likes(Harry, Ron)\}$

"at least one of these has to hold"

Refresh:Universally quantified formulas

A universally quantified formula holds in a KB, if all of its instantiations hold.

$$hasSpouse(x, y)$$

$$hasSpouse(x, y) \Rightarrow hasSpouse(y, x)$$

$$hasSpouse(Elvis, y) \Rightarrow hasSpouse(y, Elvis)$$

$$hasSpouse(Ron, y) \Rightarrow hasSpouse(y, Ron)$$

Refresh:Universally quantified formulas

In the following, we consider only instantiated rules.

A universally quantified rule is a shorthand notation for all relevant instantiations.

$$hasSpouse(x, y) \Rightarrow hasSpouse(y, x)$$

relevant instantiations

$$hasSpouse(Elvis, Priscilla) \Rightarrow hasSpouse(Priscilla, Elvis)$$
$$hasSpouse(Elvis, Elvis) \Rightarrow hasSpouse(Elvis, Elvis)$$
$$hasSpouse(Davis, Reagan) \Rightarrow hasSpouse(Reagan, Davis)$$
...

irrelevant $hasSpouse(dog, cat) \Rightarrow hasSpouse(cat, dog)$

Def: Weighted Rule

A weighted rule is a rule with an associated real-valued weight.

$$hasSpouse(Elvis, Priscilla) \Rightarrow hasSpouse(Priscilla, Elvis)[3.14]$$

A weighted rule can also be seen as a weighted disjunction...

$$\neg hasSpouse(Elvis, Priscilla) \vee hasSpouse(Priscilla, Elvis)[3.14]$$

...or a weighted clause.

$$\{\neg hasSpouse(Elvis, Priscilla), hasSpouse(Priscilla, Elvis)\}[3.14]$$

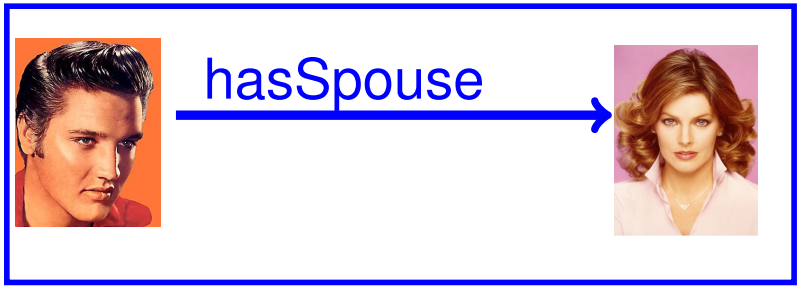
Def: Weight of a KB

Given a set of atoms (= possible world, KB) and a set of instantiated rules with weights, the weight of the KB is the sum of the weights of all rules that hold in the KB.

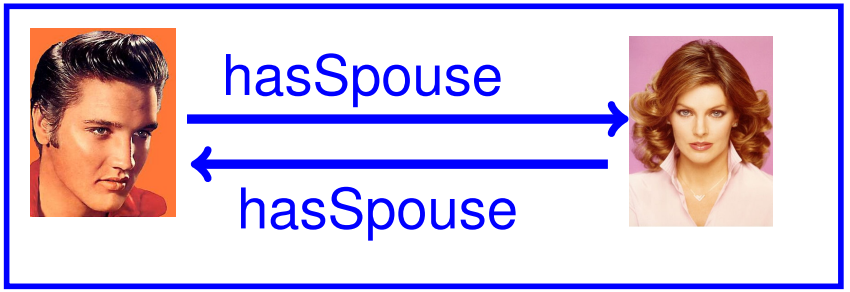
$$hasSpouse(Elvis, Priscilla) \Rightarrow hasSpouse(Priscilla, Elvis)[3]$$

$$hasSpouse(cat, dog) \Rightarrow hasSpouse(dog, cat)[2]$$

KB1



KB2



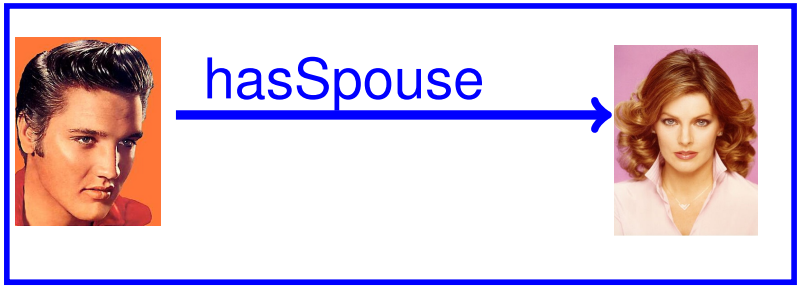
Def: Weight of a KB

Given a set of atoms (= possible world, KB) and a set of instantiated rules with weights, the weight of the KB is the sum of the weights of all rules that hold in the KB.

$$hasSpouse(Elvis, Priscilla) \Rightarrow hasSpouse(Priscilla, Elvis)[3]$$

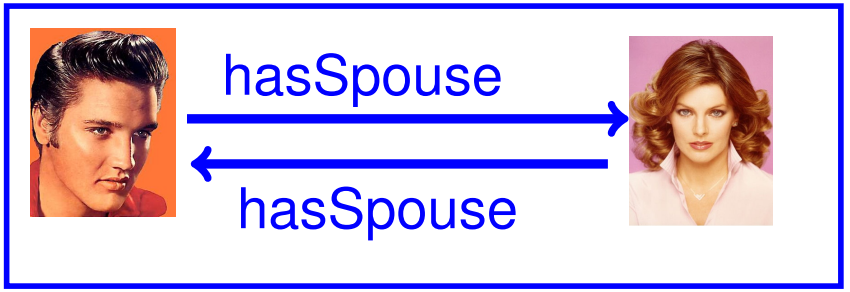
$$hasSpouse(cat, dog) \Rightarrow hasSpouse(dog, cat)[2]$$

KB1



Weight: 2

KB2



Weight: 5

Def: Weighted MAX SAT

Given a set of instantiated rules with weights, weighted MAX SAT is the problem of finding the KB with the highest weight. If there are several, find the one with the least number of atoms.

$is(Ron, immature)$[10]

$is(Ron, immature) \wedge type(H., sorceress) \Rightarrow likes(H., Ron)$[3]

$type(Hermione, sorceress)$[4]

Def: Weighted MAX SAT

Given a set of instantiated rules with weights, weighted MAX SAT is the problem of finding the KB with the highest weight. If there are several, find the one with the least number of atoms.

$is(Ron, immature)$[10]

$is(Ron, immature) \wedge type(H., sorceress) \Rightarrow likes(H., Ron)$[3]

$type(Hermione, sorceress)$[4]

Best world:

$is(Ron, immature)$

$type(Hermione, sorceress)$

$likes(Hermione, Ron)$

weight: 17

Def: Exhaustive search

Exhaustive search is an algorithm for the Weighted MAX SAT problem that tries out all possible worlds with the atoms that appear in the rules in order to find the possible world with the maximal weight.

$is(Ron, immature)$[10]

$is(Ron, immature) \wedge type(H., sorceress) \Rightarrow likes(H., Ron)$[3]

$type(Hermione, sorceress)$[4]

Atoms:

$is(Ron, immature), type(H., sorceress), likes(H.Ron)$

Def: Exhaustive search

Exhaustive search is an algorithm for the Weighted MAX SAT problem that tries out all possible worlds with the atoms that appear in the rules in order to find the possible world with the maximal weight.

$is(Ron, immature)$[10]

$is(Ron, immature) \wedge type(H., sorceress) \Rightarrow likes(H., Ron)$[3]

$type(Hermione, sorceress)$[4]

Atoms:

$is(Ron, immature), type(H., sorceress), likes(H.Ron)$

Possible worlds:

$\{\}$: weight 3

$\{is(Ron, immature)\}$: weight 13,

$\{is(Ron, immature), type(H., sorceress)\}$: weight 14, etc.

Def: Exhaustive search

Exhaustive search is an algorithm for the Weighted MAX SAT problem that tries out all possible worlds with the atoms that appear in the rules in order to find the possible world with the maximal weight.

$is(Ron, immature)$[10]

$is(Ron, immature) \wedge type(H., sorceress) \Rightarrow likes(H., Ron)$[3]

$type(Hermione, sorceress)$[4]

Atoms:

$is(Ron, immature), type(H., sorceress), likes(H.Ron)$

Exhaustive search is a correct and complete algorithm for the Weighted Max Sat problem. However, it has to analyze 2^n possible worlds, where n is the number of atoms.

Intuition of unit propagation

Unit propagation repeatedly (1) adds the head of a body-less rule to the KB, (2) removes this atom from the bodies of the other rules, and (3) removes definitively satisfied rules.

$is(Ron, immature)$[10]

$is(Ron, immature) \wedge type(H., sorceress) \Rightarrow likes(H., Ron)$[3]

$type(Hermione, sorceress)$[4]

>example&unitcl

Intuition of unit propagation

Unit propagation repeatedly (1) adds the head of a body-less rule to the KB, (2) removes this atom from the bodies of the other rules, and (3) removes definitively satisfied rules.

$is(Ron, immature)$[10]

$is(Ron, immature) \wedge type(H., sorceress) \Rightarrow likes(H., Ron)$[3]

$type(Hermione, sorceress)$[4]

KB: $\{is(Ron, immature)\}$

>example&unitcl

Intuition of unit propagation

Unit propagation repeatedly (1) adds the head of a body-less rule to the KB, (2) removes this atom from the bodies of the other rules, and (3) removes definitively satisfied rules.

~~$is(Ron, immature)$[10]~~

~~$is(Ron, immature)$~~ $\wedge\, type(H., sorceress) \Rightarrow likes(H., Ron)$[3]

$type(Hermione, sorceress)$[4]

KB: $\{is(Ron, immature)\}$

>example&unitcl

Intuition of unit propagation

Unit propagation repeatedly (1) adds the head of a body-less rule to the KB, (2) removes this atom from the bodies of the other rules, and (3) removes definitively satisfied rules.

~~$is(Ron, immature)$[10]~~

~~$is(Ron, immature)$~~ $\wedge$ ~~$type(H., sorceress)$~~ $\Rightarrow likes(H., Ron)$[3]

~~$type(Hermione, sorceress)$~~[4]

KB: $\{is(Ron, immature), type(Hermione, sorceress)\}$

>example&unitcl

Intuition of unit propagation

Unit propagation repeatedly (1) adds the head of a body-less rule to the KB, (2) removes this atom from the bodies of the other rules, and (3) removes definitively satisfied rules.

~~$is(Ron, immature)$~~[10]

~~$is(Ron, immature)$~~ $\wedge$ ~~$type(H., sorceress) \Rightarrow likes(H., Ron)$~~[3]

~~$type(Hermione, sorceress)$~~[4]

KB: $\{is(Ron, immature), type(Hermione, sorceress), likes(H., Ron)\}$

>unitcl

Unit propagation for clauses

Unit propagation repeatedly (1) adds the atom of a unit-clause to the KB, (2) removes all clauses that contain this atom, and (3) removes the negation of the atom from the other clauses.

$\{is(Ron, immature)\}$[10]

$\{\neg is(Ron, immature), \neg type(H., sorceress), likes(H., Ron)\}$[3]

$\{type(Hermione, sorceress)\}$[4]

Unit propagation for clauses

Unit propagation repeatedly (1) adds the atom of a unit-clause to the KB, (2) removes all clauses that contain this atom, and (3) removes the negation of the atom from the other clauses.

$\{is(Ron, immature)\}$[10]

$\{\neg is(Ron, immature), \neg type(H., sorceress), likes(H., Ron)\}$[3]

$\{type(Hermione, sorceress)\}$[4]

KB: $\{is(Ron, immature)\}$

Unit propagation for clauses

Unit propagation repeatedly (1) adds the atom of a unit-clause to the KB, (2) removes all clauses that contain this atom, and (3) removes the negation of the atom from the other clauses.

~~$\{is(Ron, immature)\}$~~[10]

$\{\neg is(Ron, immature), \neg type(H., sorceress), likes(H., Ron)\}$[3]

$\{type(Hermione, sorceress)\}$[4]

KB: $\{is(Ron, immature)\}$

Unit propagation for clauses

Unit propagation repeatedly (1) adds the atom of a unit-clause to the KB, (2) removes all clauses that contain this atom, and (3) removes the negation of the atom from the other clauses.

~~$\{is(Ron, immature)\}$~~[10]

~~$\neg is(Ron, immature)$~~, $\neg type(H., sorceress), likes(H., Ron)\}$[3]

$\{type(Hermione, sorceress)\}$[4]

KB: $\{is(Ron, immature)\}$

etc.

Def: Unit propagation for clauses

Unit propagation repeatedly (1) adds the atom of a unit-clause to the KB, (2) removes all clauses that contain this atom, and (3) removes the negation of the atom from the other clauses.

Wikipedia/Unit propagation with a partial model

~~$\{is(Ron, immature)\}$~~[10]

$\{$~~$\neg is(Ron, immature)$~~, $\neg type(H., sorceress), likes(H., Ron)\}$[3]

$\{type(Hermione, sorceress)\}$[4]

Unit propagation is a correct (but not complete) algorithm for the Weighted Max Sat Problem, if a unit clause is propagated only when its weight is higher than the sum of the weights of the rules where the literal appears with inverse polarity.

Task: Weighted MAX SAT

Find the KB with the highest weight:

$is(Hermione, smart)$[1]

$is(Herm., smart) \wedge is(Harry, smart) \Rightarrow likes(Herm., Harry)$[3]

$likes(Hermione, Ron) \Rightarrow \neg likes(Hermione, Harry)$[100]

$is(Harry, smart)$[10]

$likes(Hermione, Ron)$[20]

Solving Weighted MAX SAT

To always find the optimal solution, one has to do an exhaustive search. Since SAT is NP-complete, so is MAX SAT and Weighted MAX SAT.

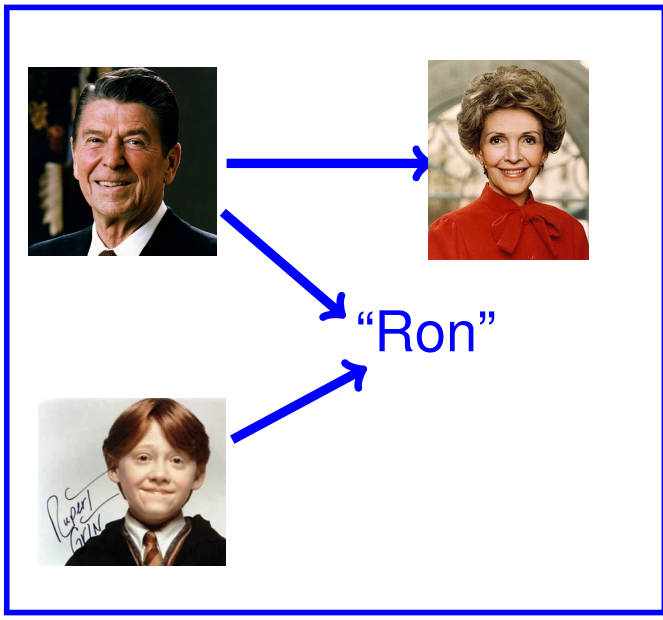
To find an approximate solution, possible strategies are:

- do an exhaustive search if there are few atoms
- try out several random KBs
- apply unit propagation wherever possible
- give preference to rules/unit clauses with higher weights
- remove atoms that appear only negative, add atoms to the KB that appear only positive.

Back to our problem

"Hermione is married to Ron"

transform everything to logical formulas



$A \wedge B \Rightarrow C$

Find best conclusion

$hasSpouse(Hermione, RonWeasley)$

Consistency

Consistency constraints can be expressed by rules:

$$hasSpouse(X, Y) \wedge different(Y, Z) \Rightarrow \neg hasSpouse(X, Z)[10]$$

$$hasSpouse(X, Y) \Rightarrow type(X, person)[20]$$

$$loves(X, Y) \wedge \neg hasSpouse(X, Y) \Rightarrow type(X, stupidPerson)[3]$$

...

Rules and weights can be designed manually. Such rules will guide our information extraction process.

A KB can be expressed as rules

Every fact from the KB can be expressed as a weighted rule:

$$type(Hermione, Person)[100]$$

High weight

This is corresponds to the rule

$$\Rightarrow type(Hermione, Person)[100]$$

Assuming completeness

If we assume the KB to be perfect on a relation, we can create negative rules:

$$\neg type(Hermione, X)[100]$$

$$\forall X : type(Hermione, X) \notin KB$$

i.e.

$$\neg type(Hermione, chicken)[100]$$

$$\neg type(Hermione, cat)[100]$$

$$\neg type(Hermione, mouse)[100]$$

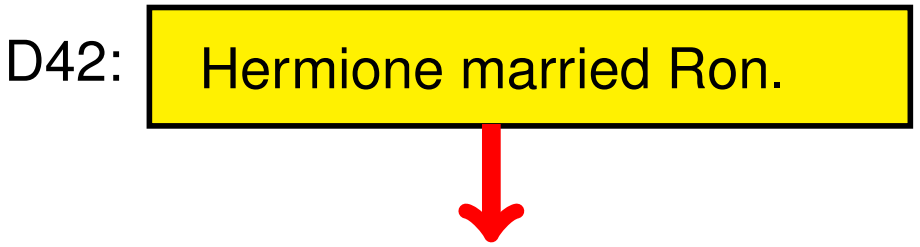
...

Expressing the corpus as rules

D42: Hermione married Ron.

$occurs$(Hermione@D42, “married”, Ron@D42)

Expressing the corpus as rules



$occurs$(Hermione@D42, “married”, Ron@D42)

Does not talk about Ronald Reagan or Ron Weasley, but about the word “Ron” in document D42.

Ron@D42 is a “word in context” (wic).

Expressing the corpus as rules

D42: Hermione married Ron.

$occurs$(Hermione@D42, “married”, Ron@D42)

The word “Ron” in document D42 can mean different entities (from KB):

$means$(Ron@D42, $RonaldReagan$)

$means$(RonD42, $RonWeasley$)

But only one entity in practice:

$$means(X, Y) \wedge different(Y, Z) \Rightarrow \neg means(X, Z)$$

Weights for corpus rules

$occurs$(Hermione@D42, “married”, Ron@D42)[3]

$means$(Ron@D42, $RonaldReagan$)

$means$(Ron@D42, $RonWeasley$)

$means(X, Y) \wedge different(Y, Z) \Rightarrow \neg means(X, Z)$

Weights for corpus rules

occurrences

$occurs$(Hermione@D42, “married”, Ron@D42)[3]

$means$(Ron@D42, $RonaldReagan$)

$means$(Ron@D42, $RonWeasley$)

$means(X, Y) \wedge different(Y, Z) \Rightarrow \neg means(X, Z)$

Weights for corpus rules

$occurs$(Hermione@D42, “married”, Ron@D42)[3]

From disambiguation by context/prior

$means$(Ron@D42, $RonaldReagan$)[5]

$means$(Ron@D42, $RonWeasley$)[7]

$$means(X, Y) \wedge different(Y, Z) \Rightarrow \neg means(X, Z)$$

Weights for corpus rules

$occurs$(Hermione@D42, “married”, Ron@D42)[3]

$means$(Ron@D42, $RonaldReagan$)[5]

$means$(Ron@D42, $RonWeasley$)[7]

“Hard” rule with very high weight

$means(X, Y) \wedge different(Y, Z) \Rightarrow \neg means(X, Z)$ [100]

Weights for corpus rules

$occurs$(Hermione@D42, “married”, Ron@D42)[3]

$means$(Ron@D42, $RonaldReagan$)

$means$(Ron@D42, $RonWeasley$)

$means(X, Y) \wedge different(Y, Z) \Rightarrow \neg means(X, Z)$ [100]

Let us ignore the weights for a moment.

Deducing patterns

KB

hasSpouse →

+

Reagan married Davis.

=

"X married Y"

is pattern for

$hasSpouse$(X,Y)

Deducing patterns

KB

hasSpouse

$hasSpouse(Reagan, Davis)$

+

Reagan married Davis.

=

"X married Y"

is pattern for

$hasSpouse$(X,Y)

Deducing patterns

KB

hasSpouse →

$hasSpouse(Reagan, Davis)$

$occurs$(R@1,“married”, D@1)

$means$(R@1, $Reagan$)

$means$(D@1, $Davis$)

+

Reagan married Davis.

=

“X married Y”

is pattern for

$hasSpouse$(X,Y)

Deducing patterns

KB

hasSpouse →

$hasSpouse(Reagan, Davis)$

+

Reagan married Davis.

$occurs$(R@1,“married”, D@1)

$means$(R@1, $Reagan$)

$means$(D@1, $Davis$)

=

“X married Y”

is pattern for

$hasSpouse$(X,Y)

$$occurs(X, P, Y) \wedge means(X, X') \wedge means(Y, Y') \wedge R(X', Y') \Rightarrow isPatternFor(P, R)$$

Deducing patterns

KB

hasSpouse →

$hasSpouse(Reagan, Davis)$

+

Reagan married Davis.

$occurs$(R@1,“married”, D@1)

$means$(R@1, $Reagan$)

$means$(D@1, $Davis$)

=

“X married Y”

is pattern for

$hasSpouse$(X,Y)

$$occurs(X, P, Y)$$
$$\wedge means(X, X')$$
$$\wedge means(Y, Y')$$
$$\wedge R(X', Y')$$
$$\Rightarrow isPatternFor(P, R)$$

$isPatternFor$(“married”, $hasSpouse$)

Applying patterns

"X married Y"

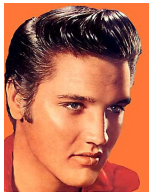
is pattern for

$hasSpouse$(X,Y)

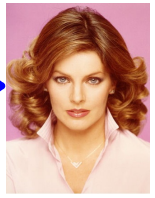
+

Elvis married Priscilla.

=



hasSpouse



Applying patterns

"X married Y"

is pattern for

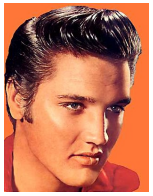
$hasSpouse$(X,Y)

$isPatternFor$("married",$hasSpouse$)

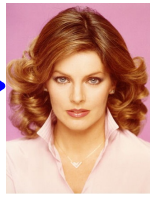
+

Elvis married Priscilla.

=



hasSpouse



Applying patterns

"X married Y"

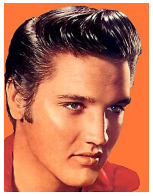
is pattern for

$hasSpouse$(X,Y)

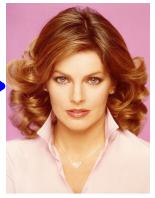
+

Elvis married Priscilla.

=



hasSpouse



$isPatternFor$("married",$hasSpouse$)

$occurs$(E@1, "married", P@1)

$means$(E@1, $Elvis$)

$means$(P@1, $Priscilla$)

Applying patterns

"X married Y"

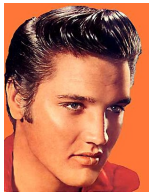
is pattern for

$hasSpouse$(X,Y)

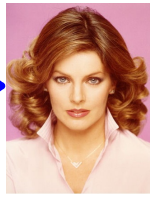
+

Elvis married Priscilla.

=



hasSpouse



$isPatternFor$("married",$hasSpouse$)

$occurs$(E@1, "married", P@1)

$means$(E@1, $Elvis$)

$means$(P@1, $Priscilla$)

$$occurs(X, P, Y)$$
$$\wedge means(X, X')$$
$$\wedge means(Y, Y')$$
$$\wedge isPatternFor(P, R)$$
$$\Rightarrow R(X', Y')$$

Applying patterns

"X married Y"

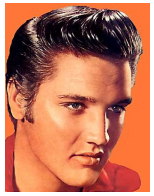
is pattern for

$hasSpouse$(X,Y)

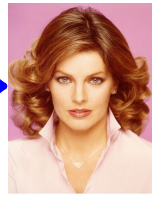
+

Elvis married Priscilla.

=



hasSpouse



$isPatternFor$("married",$hasSpouse$)

$occurs$(E@1, "married", P@1)

$means$(E@1, $Elvis$)

$means$(P@1, $Priscilla$)

$$occurs(X, P, Y)$$
$$\wedge means(X, X')$$
$$\wedge means(Y, Y')$$
$$\wedge isPatternFor(P, R)$$
$$\Rightarrow R(X', Y')$$

$$hasSpouse(Elvis, Priscilla)$$

task>106

Task: Pattern deduction by rules

Pattern deduction:

$$occurs(X, P, Y) \wedge means(X, X') \wedge means(Y, Y') \wedge R(X', Y') \Rightarrow isPatternFor(P, R)$$

Pattern application:

$$occurs(X, P, Y) \wedge means(X, X') \wedge means(Y, Y') \wedge isPatternFor(P, R) \Rightarrow R(X', Y')$$

1 : $occurs$(P@1, “adores”, E@1)

2 : $means$(E@1,$Elvis$)

3 : $means$(P@1, $Priscilla$)

4 : $hasSpouse(Priscilla, Elvis)$

5 : $occurs$(M@1, “adores”, E@1)

6 : $means$(M@1, $Madonna$)

All rules have weight 1.

Compute facts that will be in the best world.

Life is not easy

- words are ambiguous

  "Ron"

- corpora may err

  "Madonna is married to Elvis"

- parsing may fail

  "Coyote dreams Coyote eats Roadrunner"

- contradictions may occur

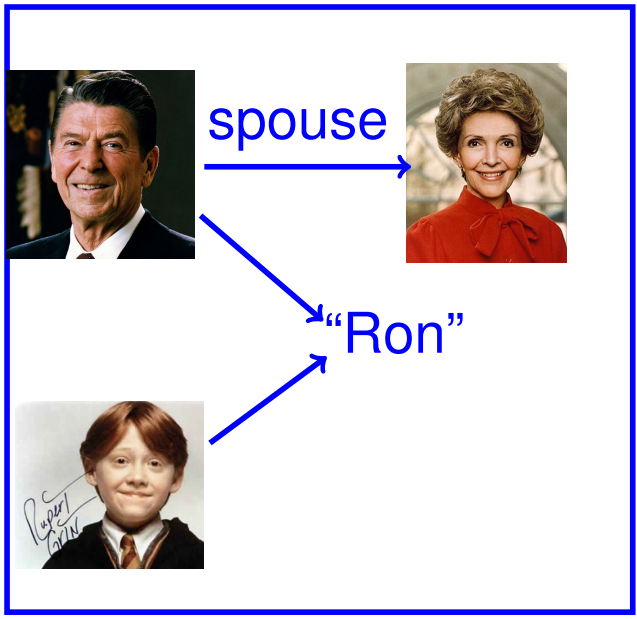
  Reagan was married twice.

$occurs$(W@1, "eats", R@1)

=> we will compute the most plausible world

Finding the most plausible world

"Hermione married Ron"



World1:

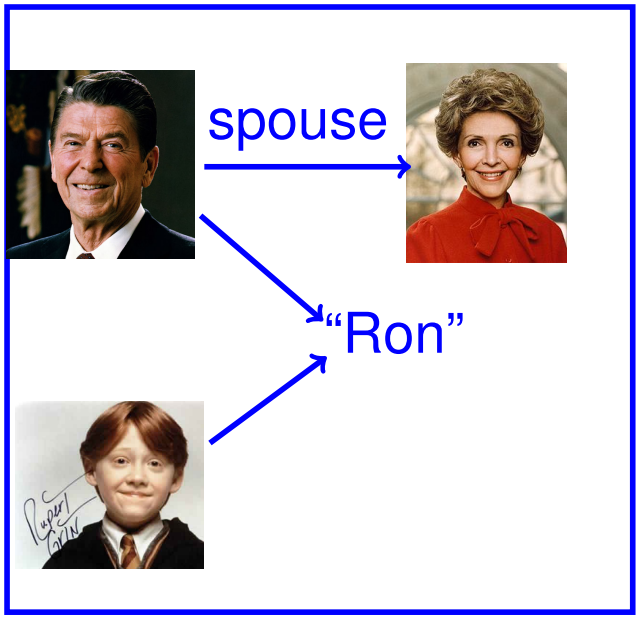
World2:

Finding the most plausible world

"Hermione married Ron"

$occurs$(H@1, "married",R@ 1)[1]

$isPatternFor$("married", $spouse$)[1]



World1:

World2:

Finding the most plausible world

"Hermione married Ron"

$occurs$(H@1, "married",R@ 1)[1]

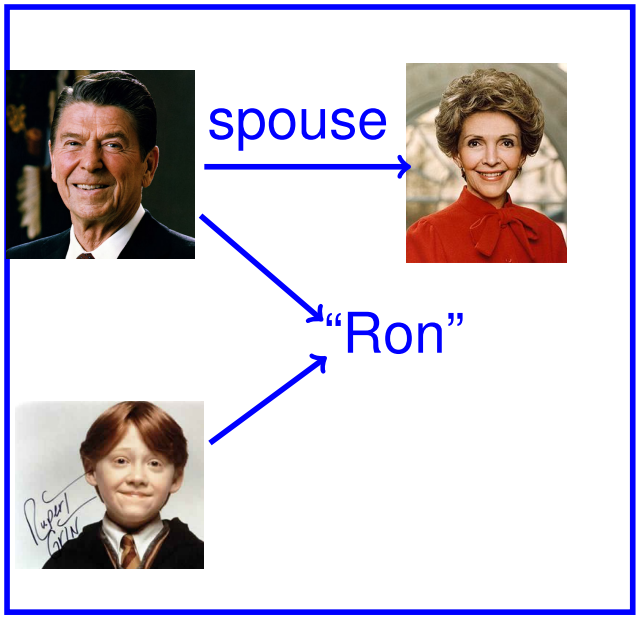
$isPatternFor$("married", $spouse$)[1]

$means$(H@1, $Hermione$)[5]

$means$(R@1, $RonWeasley$)[2]

$means$(R@1, $Reagan$)[3]

$means(X, Y) \wedge Y \neq Z \Rightarrow \neg means(X, Z)$[10]



World1:

World2:

Finding the most plausible world

"Hermione married Ron"

$occurs$(H@1, "married",R@ 1)[1]

$isPatternFor$("married", $spouse$)[1]

$means$(H@1, $Hermione$)[5]

$means$(R@1, $RonWeasley$)[2]

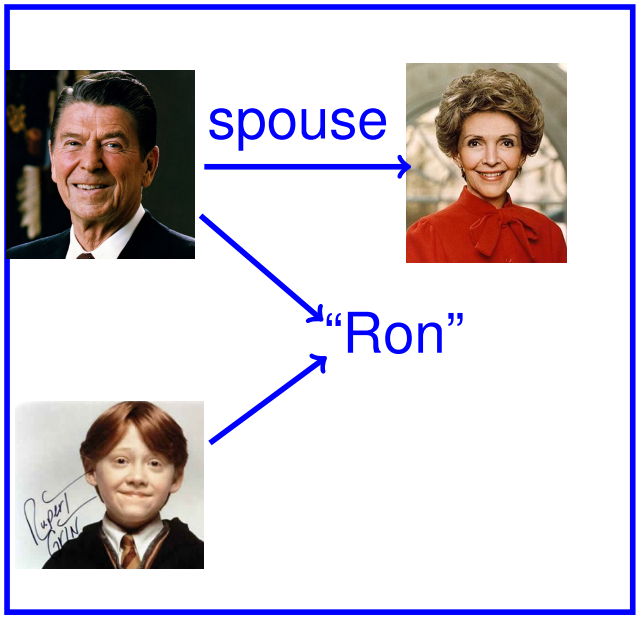
$means$(R@1, $Reagan$)[3]

$means(X, Y) \wedge Y \neq Z \Rightarrow \neg means(X, Z)$[10]

$spouse(X, Y) \wedge Y \neq Z \Rightarrow \neg spouse(Z, X)$[6]

+ Symmetry of marriage

+ Pattern Application Rule [10]



World1:

World2:

Finding the most plausible world

"Hermione married Ron"

$occurs$(H@1, "married",R@ 1)[1]

$isPatternFor$("married", $spouse$)[1]

$means$(H@1, $Hermione$)[5]

$means$(R@1, $RonWeasley$)[2]

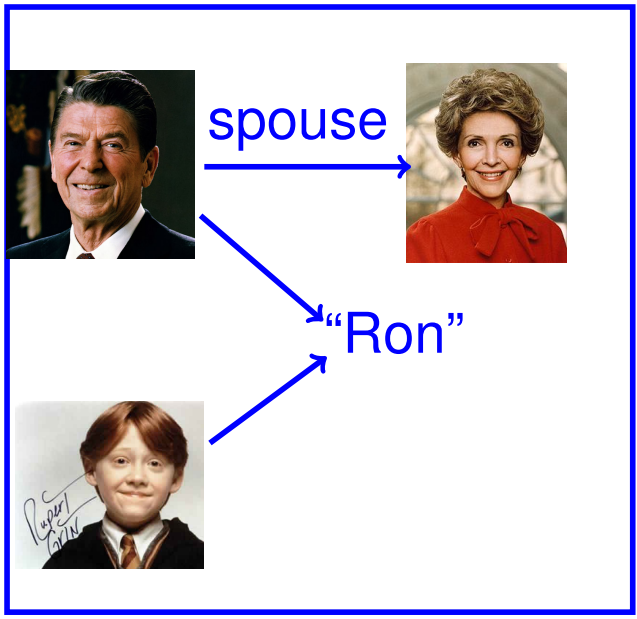
$means$(R@1, $Reagan$)[3]

$means(X, Y) \wedge Y \neq Z \Rightarrow \neg means(X, Z)$[10]

$spouse(X, Y) \wedge Y \neq Z \Rightarrow \neg spouse(Z, X)$[6]

+ Symmetry of marriage

+ Pattern Application Rule [10]



World1:

loses 2

wins 3

loses 6

World2:

Finding the most plausible world

"Hermione married Ron"

$occurs$(H@1, "married",R@ 1)[1]

$isPatternFor$("married", $spouse$)[1]

$means$(H@1, $Hermione$)[5]

$means$(R@1, $RonWeasley$)[2]

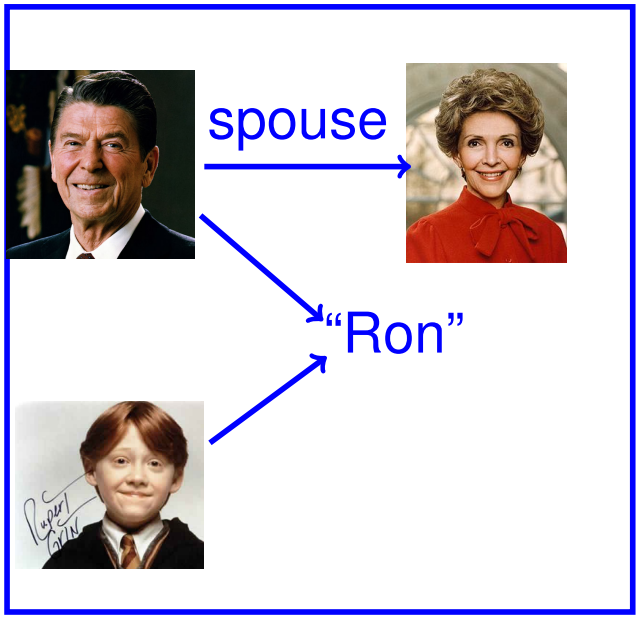
$means$(R@1, $Reagan$)[3]

$means(X, Y) \wedge Y \neq Z \Rightarrow \neg means(X, Z)$[10]

$spouse(X, Y) \wedge Y \neq Z \Rightarrow \neg spouse(Z, X)$[6]

+ Symmetry of marriage

+ Pattern Application Rule [10]



World1:

loses 2

wins 3

loses 6

World2:

loses 3

wins 6

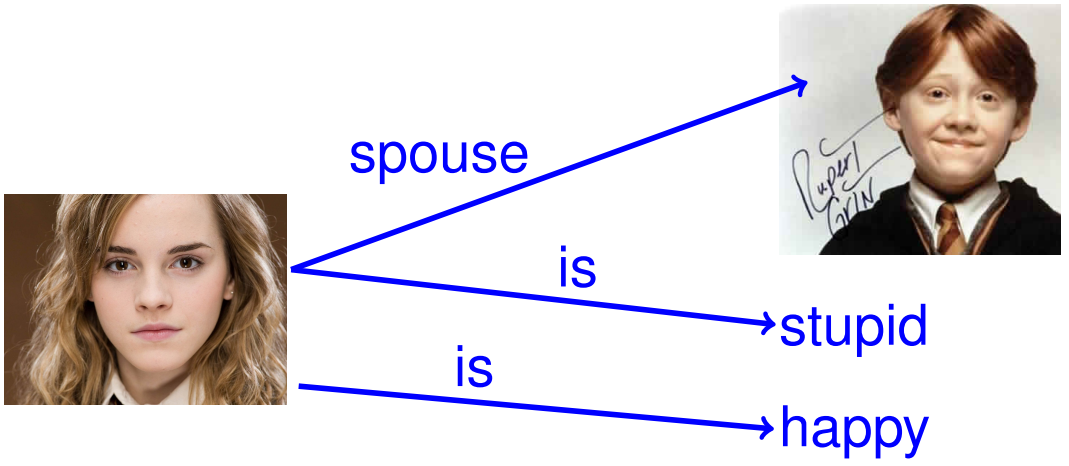
wins 2

Summary: IE by reasoning

IE by reasoning converts

- background knowledge (existing KB)
- logical constraints
- the corpus

into logical rules and aims to find the most plausible possible world given these evidences.



->markov-logic

->semantic-web

References

SOFIE - a self-organizing framework for IE

->markov-logic

->semantic-web