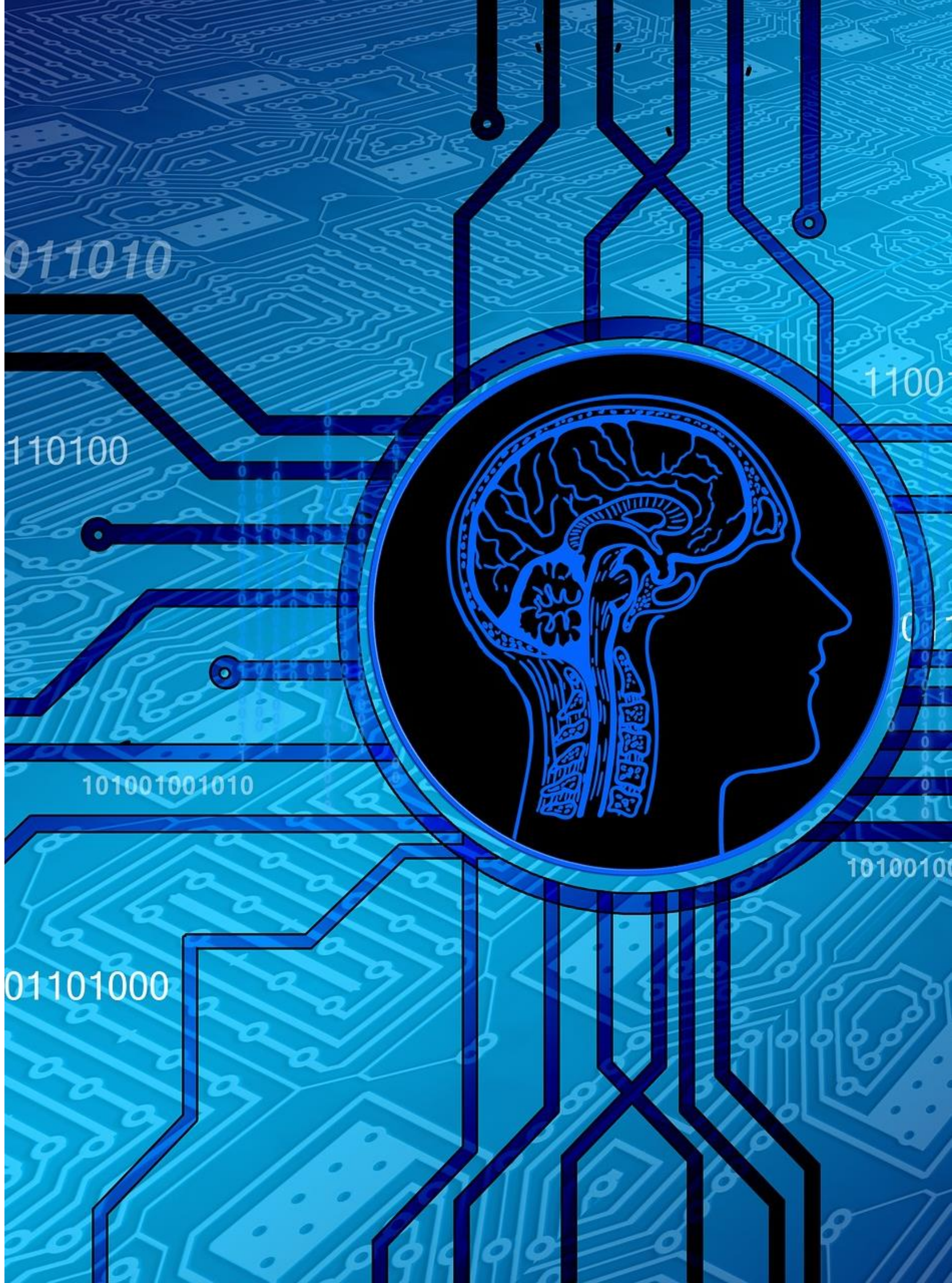
Unification & Generalised Modus Ponens

Petros Papapanagiotou

Informatics 2D: Reasoning and Agents
Lecture 12



Universal instantiation (UI)

Every instantiation of a universally quantified formula α is entailed by it:

$$\frac{\forall v.\, a}{a\{v/g\}}$$

for any variable v and ground term g

Contains no variables!

Example: $\forall x.\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ yields:

- $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
- $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
- $King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$

etc…

Existential instantiation (EI)

For any formula α, variable v, and some constant symbol k that does not appear elsewhere in the knowledge base:

$$\frac{\exists v.\, a}{a\{v/k\}}$$

Example. $\exists x.\ Crown(x) \land OnHead(x, John)$ yields:

- $Crown(C_1) \land OnHead(C_1, John)$

provided C_1 is a new constant symbol, called a Skolem constant

Reduction to propositional inference

Suppose the KB contains just the following:

$\forall x.\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ $\quad$ $King(John)$ $\quad$ $Greedy(John)$ $\quad$ $Brother(Richard, John)$

Instantiating the universal sentence in all possible ways, we have:

- $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
- $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
- $King(John)$
- $Greedy(John)$
- $Brother(Richard, John)$

Universal sentence can then be discarded!

The new KB is propositionalized: proposition symbols are

$King(John), Greedy(John), Evil(John), King(Richard)$, etc.

Not in KB as a fact!

Reduction contd.

Every FOL KB can be propositionalized so as to preserve entailment

- A ground sentence is entailed by new KB iff entailed by original KB

Idea: propositionalize KB and query, apply DPLL (or some other complete propositional method), return result

Problem: with function symbols, there are infinitely many ground terms,

- e.g., *Father*(*Father*(*Father*(*John*)))

Reduction contd.

Theorem: Herbrand (1930)

- If a sentence α is entailed by a FOL KB, it is entailed by a finite subset of the propositionalized KB

Idea: For $n = 0$ to ∞ do

- create a propositional KB by instantiating with depth-n terms
- see if α is entailed by this KB

Problem: works if α is entailed, loops forever if α is not entailed

Theorem: Turing (1936), Church (1936).

- Entailment for FOL is semi-decidable (i.e. algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence.)

Problems with propositionalization

Propositionalization seems to generate lots of irrelevant sentences.

Example:

$$\forall x.\ \mathrm{King}(x) \wedge \mathrm{Greedy}(x) \Rightarrow \mathrm{Evil}(x) \qquad \mathrm{King}(\mathrm{John})$$

$$\forall y.\ \mathrm{Greedy}(y) \qquad \mathrm{Brother}(\mathrm{Richard},\mathrm{John})$$

- It seems obvious that *Evil*(*John*), but propositionalization produces lots of facts such as *Greedy*(*Richard*) that are irrelevant.

With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations.

Unification

Unification

We can get the inference *immediately* if we can find a substitution θ such that such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*.

$$\theta = \{x/John, y/John\}$$

More generally:

$$Unify(\alpha, \beta) = \theta \iff \alpha\theta = \beta\theta$$

Unification examples

α	β	θ
Knows(John, x)	*Knows(John, Jane)*	
Knows(John, x)	*Knows(y, OJ)*	
Knows(John, x)	*Knows(y, Mother(y))*	
Knows(John, x)	*Knows(x, Richard)*	

Unification examples

α	β	θ
Knows(John, x)	*Knows(John, Jane)*	*{x/Jane}*
Knows(John, x)	*Knows(y, OJ)*	
Knows(John, x)	*Knows(y, Mother(y))*	
Knows(John, x)	*Knows(x, Richard)*	

Unification examples

α	β	θ
Knows(John, x)	*Knows(John, Jane)*	*{x/Jane}*
Knows(John, x)	*Knows(y, OJ)*	*{x/OJ, y/John}*
Knows(John, x)	*Knows(y, Mother(y))*	
Knows(John, x)	*Knows(x, Richard)*	

Unification examples

α	β	θ
Knows(John, x)	*Knows(John, Jane)*	$\{x/Jane\}$
Knows(John, x)	*Knows(y, OJ)*	$\{x/OJ, y/John\}$
Knows(John, x)	*Knows(y, Mother(y))*	$\{y/John, x/Mother(John)\}$
Knows(John, x)	*Knows(x, Richard)*	

Unification examples

α	β	θ
Knows(John, x)	*Knows(John, Jane)*	*{x/Jane}*
Knows(John, x)	*Knows(y, OJ)*	*{x/OJ, y/John}*
Knows(John, x)	*Knows(y, Mother(y))*	*{y/John, x/Mother(John)}*
Knows(John, x)	*Knows(x, Richard)*	**Fail!**

Unification examples

α	β	θ
Knows(John, x)	*Knows(John, Jane)*	*{x/Jane}*
Knows(John, x)	*Knows(y, OJ)*	*{x/OJ, y/John}*
Knows(John, x)	*Knows(y, Mother(y))*	*{y/John, x/Mother(John)}*
Knows(John, x)	*Knows(x, Richard)*	**Fail!**

Standardizing variables apart eliminates overlap of variables

e.g. change *Knows(x, Richard)* to *Knows*(z_{17}, *Richard)* and then we succeed the last case with

$\theta = \{z_{17}/John, x/Richard\}$

MGU

Unifying *Knows(John, x)* and *Knows(y, z)*

$$\theta = \{y/John, x/z\} \quad \text{or} \quad \theta = \{y/John, x/John, z/John\}$$

The first unifier is more general than the second.

FOL: There is a **single** most general unifier (MGU) that is unique up to renaming of variables.

$$\text{MGU} = \{y/John, x/z\}$$

Can be viewed as an equation solving problem.

- *i.e. solve* $Knows(John, x) \stackrel{?}{=} Knows(y, z)$

MGU Examples

	MGU
$Loves(John, x) \stackrel{?}{=} Loves(y, Mother(y))$	
$Loves(John, Mother(y)) \stackrel{?}{=} Loves(y, y)$	

MGU Examples

	MGU
$Loves(John, x) \stackrel{?}{=} Loves(y, Mother(y))$	$\{x/Mother(John), y/John\}$
$Loves(John, Mother(y)) \stackrel{?}{=} Loves(y, y)$	

MGU Examples

	MGU
$Loves(John, x) \stackrel{?}{=} Loves(y, Mother(y))$	$\{x/Mother(John), y/John\}$
$Loves(John, Mother(y)) \stackrel{?}{=} Loves(y, y)$	Fail!

Finding the MGU

Can be broken-down into a series of steps

- Decomposition
- Conflict
- Eliminate
- Delete
- Switch
- Coalesce
- Occurs Check

Other presentations of algorithm are possible (see R&N)

Decomposition

Given

$$f(s_1, \ldots, s_n) \stackrel{?}{=} f(t_1, \ldots, t_n)$$

Replace with

$$s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n$$

Example

Given

$$Knows(John, x) \stackrel{?}{=} Knows(y, z)$$

Replace with

$$John \stackrel{?}{=} y, \quad x \stackrel{?}{=} z$$

Conflict

Given

$f(s_1, \ldots, s_n) \stackrel{?}{=} g(t_1, \ldots, t_n)$ where $f \neq g$

Fail!

Example

Given

$Knows(John, x) \stackrel{?}{=} Greedy(y)$

fail

Eliminate

Given

$P, \quad x \stackrel{?}{=} t$ where x occurs in P but not in t, and t is not a variable

Replace with

$P\{x/t\}$ and $x \stackrel{?}{=} t$

Example

Given

$Knows(John, x) \stackrel{?}{=} Knows(y, z), \quad z \stackrel{?}{=} Richard$

Replace with

$Knows(John, x) \stackrel{?}{=} Knows(y, Richard), \quad z \stackrel{?}{=} Richard$

Given

$P, \; s \stackrel{?}{=} s$

Replace with

P

Delete

Example

Given

$z \stackrel{?}{=} Richard, \quad Greedy(John) \stackrel{?}{=} Greedy(John)$

Replace with

$z \stackrel{?}{=} Richard$

Switch

Given

$P,\ s \stackrel{?}{=} x$ where x is a variable and s is not

Replace with

P and $x \stackrel{?}{=} s$

Example

Given

$Knows(John, x) \stackrel{?}{=} Knows(y, z),\quad Richard \stackrel{?}{=} z$

Replace with

$Knows(John, x) \stackrel{?}{=} Knows(y, z),\quad z \stackrel{?}{=} Richard$

Coalesce

Given

$P,\ x \stackrel{?}{=} y$ where x, y variables occurring in P

Replace with

$P\{x/y\}$ and $x \stackrel{?}{=} y$

Example

Given

$Knows(John, x) \stackrel{?}{=} Knows(y, z), \quad y \stackrel{?}{=} z$

Replace with

$Knows(John, x) \stackrel{?}{=} Knows(z, z), \quad y \stackrel{?}{=} z$

Occurs Check

Given

$x \stackrel{?}{=} s$ where x **occurs** in s and s not a variable

Fail!

Example

Given

$P(x), \quad x \stackrel{?}{=} Father(x)$

Fail (else Eliminate will loop)

P(Father(Father(Father(…))))

Example

$$Loves(John, x) \stackrel{?}{=} Loves(y, Mother(y))$$

Decompose

$$John \stackrel{?}{=} y,\ \ x \stackrel{?}{=} Mother(y)$$

Switch

$$y \stackrel{?}{=} John,\ \ x \stackrel{?}{=} Mother(y)$$

Eliminate

$$y \stackrel{?}{=} John,\ \ x \stackrel{?}{=} Mother(John)$$

Generalised Modus Ponens

Modus Ponendo Ponens

Latin for *"method of putting by placing"* – *"way that affirms by affirming"*

$$\boxed{\frac{P \qquad P \Rightarrow Q}{Q}}$$

$$P, \qquad P \Rightarrow Q \;\vdash Q$$

Generalized Modus Ponens (GMP)

$$\frac{p_1', p_2', \ldots, p_n' \quad (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta} \quad where\ p_i'\theta \equiv p_i\theta$$

Example: $\forall x.$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)

p_1' is *King*(*John*)	p_1 is *King*(*x*)
p_2' is *Greedy*(*y*)	p_2 is *Greedy*(*x*)
θ is {x/John, y/John}	q is *Evil*(*x*)
$q\theta$ is *Evil*(*John*)	

GMP used with KB of definite clauses (exactly one positive literal)

All variables assumed universally quantified

Soundness of GMP

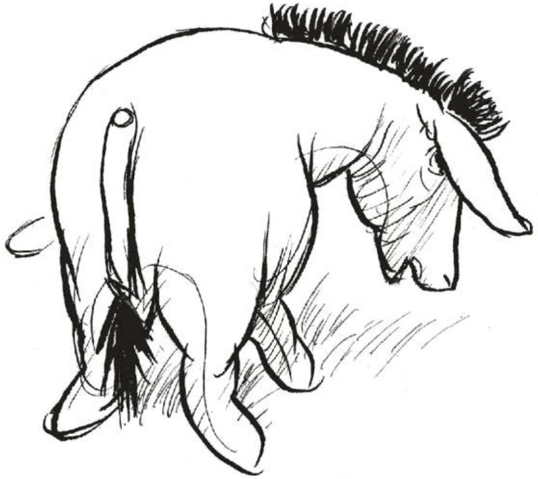
Need to show that

$$p_1', p_2', \ldots, p_n', (p_1 \land p_2 \land \ \ldots \land p_n \Rightarrow q) \vDash q\theta$$

provided that $p_i'\theta = p_i\theta$ for all *i*

Lemma: For any sentence *p*, we have $p \vDash p\theta$ by UI

1. $(p_1 \land p_2 \land \ \ldots \land p_n \Rightarrow q) \vDash (p_1 \land p_2 \land \ \ldots \land p_n \Rightarrow q)\theta = (p_1\theta \land p_2\theta \land \ \ldots \land p_n\theta \Rightarrow q\theta)$
2. $p_1', p_2', \ldots, p_n' \vDash p_1' \land p_2' \land \ \ldots \ \land p_n' \vDash (p_1' \land p_2' \land \ \ldots \ \land p_n')\,\theta$
   $= p_1'\theta \land \ p_2'\theta \land \ \ldots \ \land p_n'\theta = p_1\theta \land p_2\theta \land \ \ldots \land p_n$
3. *From 1 and 2, $q\theta$ follows by ordinary Modus Ponens*



New Example KB

Example Knowledge Base

It is known in The Hundred-Acre Wood that if someone who is very fond of food gives a treat to one of their friends, they are really generous.

Eeyore, the sad donkey, has some hunny that he has received for his birthday from Winnie-the-Pooh, who, as we know, is very fond of food.

Prove that Winnie-the-Pooh is generous.

Formalisation

if someone who is very fond of food gives a treat to one of their friends, they are really generous

- $VeryFondOfFood(x) \wedge Treat(y) \wedge Friend(z) \wedge Gives(x, y, z) \Rightarrow Generous(x)$

Eeyore (…) has some hunny

- $\exists x. Owns(Eeyore, x) \wedge Hunny(x)$ or after EI: $Owns(Eeyore, H_1) \wedge Hunny(H_1)$

that he has received for his birthday from Winnie-the-Pooh

- $Hunny(x) \wedge Owns(Eeyore, x) \Rightarrow Gives(Pooh, x, Eeyore)$

Hunny is a treat.

- $Hunny(x) \Rightarrow Treat(x)$

Residents of the the Hundred-Acre Wood are friends.

- $Resident(x, HundredAcreWood) \Rightarrow Friend(x)$

Eeyore is a resident of the the Hundred-Acre Wood.

- $Resident(Eeyore, HundredAcreWood)$

Pooh is very fond of food.

- $VeryFondOfFood(Pooh)$

Why?

Setting the scene for inference & resolution.

Linked to logic programming.

Meta-theory.

…but more in the next lecture!