The Distributed Ontology, Model and Specification Language (DOL) Day 2: Basic Structuring with DOL

Oliver Kutz[1]
Till Mossakowski[2]

[1]Free University of Bozen-Bolzano, Italy
[2]University of Magdeburg, Germany

Tutorial at ESSLLI 2016, Bozen-Bolzano, August 15 – 19

Summary of Day 1

On Day 1 we have:

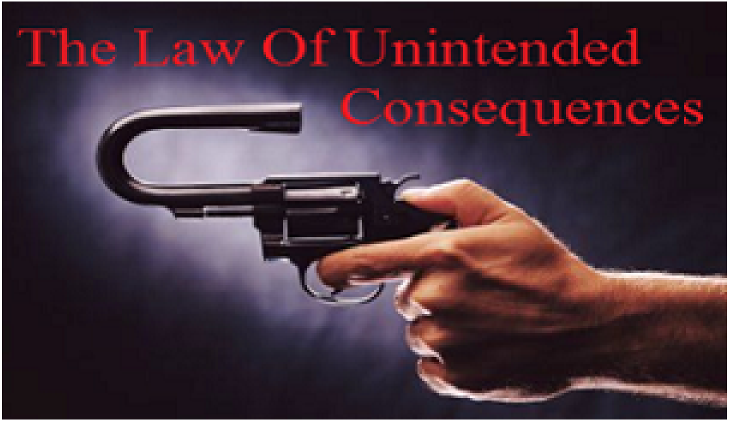
- Explored the motivation behind DOL looking at several use-cases from ontology engineering
- Introduced the basic ideas and features of DOL
- Introduced some logics we will use during the week
- Introduced the tools to be used: Ontohub and HETS

Today

We will focus today on discussing in parallel use cases for all three logics and giving DOL syntax and semantics for:

- intended consequences (competency questions)
- model finding and refutation of lemmas
- extensions and conservative extensions
- signature morphisms and the satisfaction condition
- refinements / theory interpretations

Intended Consequences



Logical Consequence in Prop, FOL and OWL

Logic deals with what follows from what.

J.A. Robinson: Logic, Form and Function.

Logical consequence = Satisfaction in a model is preserved:

$$\varphi_1, \dots, \varphi_n \models \psi$$

All models of the premises $\varphi_1, \dots, \varphi_n$
are models of the conclusion ψ.

Formally: $M \models \varphi_1$ and ... and $M \models \varphi_n$ together imply $M \models \psi$.

More general form:

$$\Phi \models \psi \qquad (\Phi \text{ may be infinite})$$

$M \models \varphi$ for all $\varphi \in \Phi$ implies $M \models \psi$.

Countermodels in Prop, FOL and OWL

Given a question about logical consequence over Σ-sentences,

$$\Phi \overset{?}{\models} \psi$$

a countermodel is a Σ-model M with

$$M \models \Phi \text{ and } M \not\models \psi$$

A countermodel shows that $\Phi \models \psi$ does not hold.


Do you think we should bite?
ATP
Proof
Model
T
P
T
P

Intended Consequences in Propositional Logic

```
logic Propositional
spec JohnMary =
 props sunny, weekend, john_tennis, mary_shopping,
       saturday %% declaration of signature
 . sunny /\ weekend => john_tennis %(when_tennis)%
 . john_tennis => mary_shopping   %(when_shopping)%
 . saturday                   %(it_is_saturday)%
 . sunny                      %(it_is_sunny)%
 . mary_shopping   %(mary_goes_shopping)% %implied
end
```

Full specification at
https://ontohub.org/esslli-2016/Propositional/leisure_structured.dol

A Countermodel

```
logic Propositional
spec Countermodel =
 props sunny, weekend, john_tennis, mary_shopping,
       saturday %% declaration of signature
  . sunny
  . not weekend
  . not john_tennis
  . not mary_shopping
  . saturday
end
```

This specification has exactly one model, and hence can be seen as a syntactic description of this model.

Repaired Specification

```
logic Propositional
spec JohnMary =
 props sunny, weekend, john_tennis, mary_shopping,
       saturday %% declaration of signature
 . sunny /\ weekend => john_tennis %(when_tennis)%
 . john_tennis => mary_shopping   %(when_shopping)%
 . saturday                  %(it_is_saturday)%
 . sunny                     %(it_is_sunny)%
 . saturday => weekend %(sat_weekend)%
 . mary_shopping   %(mary_goes_shopping)% %implied
end
```

Intended Consequences in FOL

```
logic CASL.FOL=
spec BooleanAlgebra =
  sort Elem
  ops 0,1 : Elem;
      __ cap __ : Elem * Elem -> Elem, assoc, comm, unit 1;
      __ cup __ : Elem * Elem -> Elem, assoc, comm, unit 0;
  forall x,y,z:Elem
  . x cap (x cup y) = x     %(absorption_def1)%
  . x cup (x cap y) = x     %(absorption_def2)%
  . x cap 0 = 0             %(zeroAndCap)%
  . x cup 1 = 1             %(oneAndCup)%
  . x cap (y cup z) = (x cap y) cup (x cap z)
                            %(distr1_BooleanAlgebra)%
  . x cup (y cap z) = (x cup y) cap (x cup z)
                            %(distr2_BooleanAlgebra)%
  . exists x' : Elem . x cup x' = 1 /\ x cap x' = 0
                            %(inverse_BooleanAlgebra)%
  . x cup x = x             %(idem_cup)% %implied
  . x cap x = x             %(idem_cap)% %implied
end
```

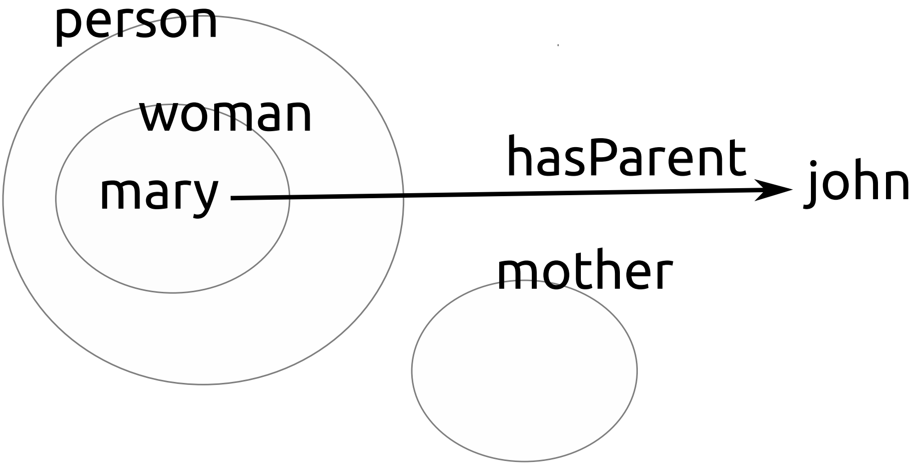
https://ontohub.org/esslli-2016/FOL/OrderTheory_structured.dol

Intended Consequences in OWL

```
logic OWL
ontology Family1 =
  Class: Person
  Class: Woman SubClassOf: Person
  ObjectProperty: hasChild
  Class: Mother
        EquivalentTo: Woman and hasChild some Person
  Individual: mary Types: Woman Facts: hasChild  john
  Individual: john
  Individual: mary
       Types: Annotations: Implied "true"^^xsd:boolean
              Mother
end
```

https://ontohub.org/esslli-2016/OWL/Family_structured.dol

A Countermodel



Repaired Ontology

```
logic OWL
ontology Family2 =
  Class: Person
  Class: Woman SubClassOf: Person
  ObjectProperty: hasChild
  Class: Mother
         EquivalentTo: Woman and hasChild some Person
  Individual: mary Types: Woman Facts: hasChild  john
  Individual: john Types: Person
  Individual: mary
       Types: Annotations: Implied "true"^^xsd:boolean
              Mother
end
```

Extensions

Structuring Using Extensions

```
logic Propositional
spec JohnMary_TBox = %% general rules
  props sunny, weekend, john_tennis, mary_shopping,
        saturday   %% declaration of signature
  . sunny /\ weekend => john_tennis %(when_tennis)%
  . john_tennis => mary_shopping   %(when_shopping)%
  . saturday => weekend  %(sat_weekend)%
end
spec JohnMary_ABox = %% specific facts
  JohnMary_TBox then
  . saturday                %(it_is_saturday)%
  . sunny                   %(it_is_sunny)%
  . mary_shopping    %(mary_goes_shopping)% %implied
end
```

Implied Extensions in Prop

```
logic Propositional
spec JohnMary_variant =
  props sunny, weekend, john_tennis, mary_shopping,
        saturday   %% declaration of signature
  . sunny /\ weekend => john_tennis %(when_tennis)%
  . john_tennis => mary_shopping   %(when_shopping)%
  . saturday => weekend  %(sat_weekend)%
then
  . saturday                    %(it_is_saturday)%
  . sunny                       %(it_is_sunny)%
then %implies
  . mary_shopping    %(mary_goes_shopping)%
end
```

Implied Extensions in OWL

```
ontology Family1 =
  Class: Person
  Class: Woman SubClassOf: Person
  ObjectProperty: hasChild
  Class: Mother
     EquivalentTo: Woman and hasChild some Person
  Individual: john Types: Person
  Individual: mary Types: Woman Facts: hasChild john
then %implies
  Individual: mary Types: Mother
end
```

Conservative Extensions in Prop

```
logic Propositional
spec Animals =
  props bird, penguin, living
  . penguin => bird
  . bird => living
then %cons
  prop animal
  . bird => animal
  . animal => living
end
```

In the extension, no “new” facts about the “old” signature follow.

A Non-Conservative Extension

```
spec Animals =
  props bird, penguin, living
  . penguin => bird
then %% not a conservative extension
  prop animal
  . bird => animal
  . animal => living
end
```

In the extension, “new” facts about the “old” signature follow, namely

```
  . bird => living
```

A Conservative Extension in FOL

```
logic CASL.FOL=
spec PartialOrder =
  sort Elem
  pred __leq__ : Elem * Elem
  . forall x:Elem. x leq x %(refl)%
  . forall x,y:Elem. x leq y /\ y leq x => x = y %(antisym)%
  . forall x,y,z:Elem. x leq y /\ y leq z => x leq z
                                                    %(trans)%
end
spec TotalOrder = PartialOrder then
  . forall x,y:Elem. x leq y \/ y leq x          %(dichotomy)%
then %cons
  pred __ < __ : Elem * Elem
  . forall x,y:Elem. x < y <=> (x leq y /\ not x = y)
                                                    %(<-def)%
end
```

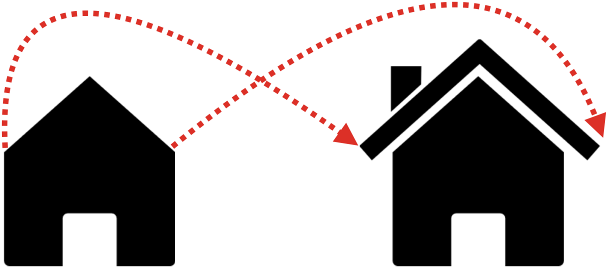
A Conservative Extension in OWL

```
logic OWL
ontology Animals1 =
  Class: LivingBeing
  Class: Bird SubClassOf: LivingBeing
  Class: Penguin SubClassOf: Bird
then %cons
  Class: Animal SubClassOf: LivingBeing
  Class: Bird SubClassOf: Animal
end
```

A Nonconservative Extension in OWL

```
logic OWL
ontology Animals2 =
  Class: LivingBeing
  Class: Bird
  Class: Penguin SubClassOf: Bird
then %% not a conservative extension
  Class: Animal SubClassOf: LivingBeing
  Class: Bird SubClassOf: Animal
end
```

Signature Morphisms and the Satisfaction Condition



Signature morphisms in propositional logic

Definition

Given two propositional signatures Σ_1, Σ_2 a signature morphism is a function $\sigma : \Sigma_1 \to \Sigma_2$. (Note that signatures are sets.)

Definition

A signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ induces a sentence translation $\mathsf{Sen}(\Sigma_1) \to \mathsf{Sen}(\Sigma_2)$, by abuse of notation also denoted by σ, defined inductively by

- $\sigma(p) = \sigma(p)$ (the two σs are different...)
- $\sigma(\bot) = \bot$
- $\sigma(\top) = \top$
- $\sigma(\phi_1 \wedge \phi_2) = \sigma(\phi_1) \wedge \sigma(\phi_2)$
- etc.

Model reduction in propositional logic

Definition

A signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ induces a model reduction function

$$\_|_\sigma : \mathrm{Mod}(\Sigma_2) \to \mathrm{Mod}(\Sigma_1).$$

Given $M \in \mathrm{Mod}(\Sigma_2)$ i.e. $M : \Sigma_2 \to \{T, F\}$,
then $M|_\sigma \in \mathrm{Mod}(\Sigma_1)$ is defined as

$$M|_\sigma(p) := M(\sigma(p))$$

for all $p \in \Sigma_1$, i.e.

$$M|_\sigma = M \circ \sigma$$

If $M'|_\sigma = M$, then M' is called a σ-expansion of M.

Satisfaction condition in propositional logic

Theorem (Satisfaction condition)

Given a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, $M_2 \in Mod(\Sigma_2)$ *and* $\phi_1 \in Sen(\Sigma_1)$, *then:*

$$M_2 \models_{\Sigma_2} \sigma(\phi_1) \textit{ iff } M_2|_\sigma \models_{\Sigma_1} \phi_1$$

(“truth is invariant under change of notation.“)

Proof.

By induction on ϕ_1. □

Signature Morphisms in FOL

Definition

Given signatures $\Sigma = (S, F, P), \Sigma' = (S', F', P')$ a signature morphism $\sigma : \Sigma \to \Sigma'$ consists of

- a map $\sigma^S : S \to S'$
- a map $\sigma^F_{w,s} : F_{w,s} \to F'_{\sigma^S(w),\sigma^S(s)}$ for each $w \in S^*$ and each $s \in S$
- a map $\sigma^P_w : P_w \to P'_{\sigma^S(w)}$ for each $w \in S^*$

Model Reduction in FOL

Definition

Given a signature morphism $\sigma : \Sigma \to \Sigma'$ and a Σ'-model M', define $M = M'|_\sigma$ as

- $M_s = M'_{\sigma^S(s)}$
- $f^M_{w,s} = \sigma^F_{w,s}(f)^{M'}_{\sigma^S(w),\sigma^S(s)}$
- $p^M_{w,s} = \sigma^P_w(p)^{M'}_{\sigma^S(w)}$

Sentence Translation in FOL

Definition

Given a signature morphism $\sigma : \Sigma \to \Sigma'$ and $\phi \in \mathrm{Sen}(\Sigma)$ the translation $\sigma(\phi)$ is defined inductively by:

$$
\begin{aligned}
\sigma(f_{w,s}(t_1 \dots t_n)) =& \sigma^F_{w,s}(f_{\sigma(w),\sigma(s))}(\sigma(t_1) \dots \sigma(t_n)) \\
\sigma(t_1 = t_2) =& \sigma(t_1) = \sigma(t_2) \\
\sigma(p_w(t_1 \dots t_n)) =& \sigma^P_w(p)_{\sigma^S(w)}(\sigma(t_1) \dots \sigma(t_n)) \\
\sigma(\phi_1 \wedge \phi_2) =& \sigma(\phi_1) \wedge \sigma(\phi_2) \qquad \text{etc.} \\
\sigma(\forall x : s.\phi) =& \forall x : \sigma^S(s).(\sigma \uplus x)(\phi) \\
\sigma(\exists x : s.\phi) =& \exists x : \sigma^S(s).(\sigma \uplus x)(\phi)
\end{aligned}
$$

where $(\sigma \uplus x) : \Sigma \uplus \{x : s\} \to \Sigma' \uplus \{x : \sigma(s)\}$ acts like σ on Σ and maps $x : s$ to $x : \sigma(s)$.

First-order Logic in DOL: Satisfaction Revisited

Definition (Satisfaction of sentences)

$M \models t_1 = t_2$ iff $M(t_1) = M(t_2)$

$M \models p_w(t_1 \dots t_n)$ iff $(M(t_1), \dots M(t_n)) \in p_w^M$

$M \models \phi_1 \wedge \phi_2$ iff $M \models \phi_1$ and $M \models \phi_2$

$M \models \forall x : s.\phi$ iff for all ι-expansions M' of M, $M' \models \phi$

where $\iota : \Sigma \hookrightarrow \Sigma \uplus \{x : s\}$ is the inclusion.

$M \models \exists x : s.\phi$ iff there is a ι-expansion M' of M such that $M' \models \phi$

Satisfaction Condition in FOL

Theorem (satisfaction condition)

For a signature morphism $\sigma : \Sigma \to \Sigma', \phi \in Sen(\Sigma), M' \in Mod(\Sigma')$*:*

$$M'|_\sigma \models \phi \textit{ iff } M' \models \sigma(\phi)$$

Proof.

For terms, prove $M'|_\sigma(t) = M'(\sigma(t))$. Then use induction on ϕ. For quantifiers, use a bijective correspondence between ι-expansions M_1 of $M'|_\sigma$ and ι'-expansions M'_1 of M'.

$$\begin{tikzcd}
M'|_\sigma & \Sigma \arrow[r, "\sigma"] \arrow[d, hook, "\iota"] & \Sigma' \arrow[d, hook, "\iota'"] & M' \\
M_1 & \Sigma \uplus \{x : s\} = \Sigma_1 \arrow[r, "\sigma \uplus x"] & \Sigma'_1 = \Sigma' \uplus \{x : \sigma(s)\} & M'_1
\end{tikzcd}$$

Signature Morphisms in OWL

Definition

Given two DL signatures $\Sigma_1 = (\mathbf{C}_1, \mathbf{R}_1, \mathbf{I}_1)$ and $\Sigma_2 = (\mathbf{C}_2, \mathbf{R}_2, \mathbf{I}_2)$ a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ consists of three functions

- $\sigma^C : \mathbf{C}_1 \to \mathbf{C}_2$,
- $\sigma^R : \mathbf{R}_1 \to \mathbf{R}_2$,
- $\sigma^I : \mathbf{I}_1 \to \mathbf{I}_2$.

Sentence Translation in OWL

Definition

Given a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ and a Σ_1-sentence ϕ, the translation $\sigma(\phi)$ is defined by inductively replacing the symbols in ϕ along σ.

Model Reduction in OWL

Definition

Given a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ and a Σ_2-model $\mathcal{I}_2$, the σ-reduct of $\mathcal{I}_2$ along σ is the Σ_1-model $\mathcal{I}_1 = \mathcal{I}_2|_\sigma$ defined by

- $\Delta^{\mathcal{I}_1} = \Delta^{\mathcal{I}_2}$
- $A^{\mathcal{I}_1} = \sigma^C(A)^{\mathcal{I}_2}, \quad$ for $A \in \mathbf{C}_1$
- $R^{\mathcal{I}_1} = \sigma^R(R)^{\mathcal{I}_2}, \quad$ for $R \in \mathbf{R}_1$
- $a^{\mathcal{I}_1} = \sigma^I(a)^{\mathcal{I}_2}, \quad$ for $a \in \mathbf{I}_1$

Satisfaction Condition in OWL

Theorem (satisfaction condition)

Given $\sigma : \Sigma_1 \to \Sigma_2$, $\phi_1 \in Sen(\Sigma_1)$ *and* $\mathcal{I}_2 \in Mod(\Sigma_2)$,

$$\mathcal{I}_2|_\sigma \models \phi_1 \quad \textit{iff} \quad \mathcal{I}_2 \models \sigma(\phi_1)$$

Proof.

Let $\mathcal{I}_1 = \mathcal{I}_2|_\sigma$. Note that $\mathcal{I}_1$ and $\mathcal{I}_2$ share the universe: $\Delta^{\mathcal{I}_1} = \Delta^{\mathcal{I}_2}$. First prove by induction over concepts C that

$$C^{\mathcal{I}_1} = \sigma(C)^{\mathcal{I}_2}.$$

Then the satisfaction condition follows easily. □

Theory Morphisms in Prop, FOL, OWL

Definition

A theory morphism $\sigma : (\Sigma_1, \Gamma_1) \to (\Sigma_2, \Gamma_2)$ is a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ such that

$$\text{for } M \in \mathrm{Mod}(\Sigma_2, \Gamma_2), \text{ we have } M|_\sigma \in \mathrm{Mod}(\Sigma_1, \Gamma_1)$$

Extensions are theory morphisms:

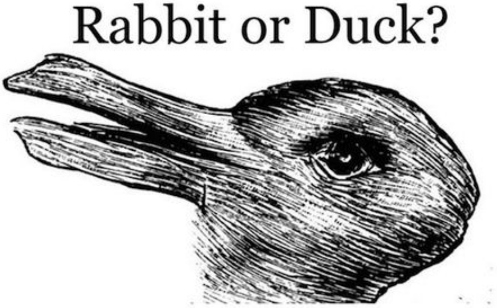
$$(\Sigma, \Gamma) \ \textbf{then}\ (\Delta_\Sigma, \Delta_\Gamma)$$

leads to the theory morphism

$$(\Sigma, \Gamma) \xrightarrow{\iota} (\Sigma \cup \Delta_\Sigma, \iota(\Gamma) \cup \Delta_\Gamma)$$

Proof: $M \models \iota(\Gamma) \cup \Delta_\Gamma$ implies $M|_\iota \models \Gamma$ by the satisfaction condition.

Interpretations



Interpretations (views, refinements)

- **interpretation** *name* : O_1 **to** $O_2 = \sigma$
- σ is a signature morphism (if omitted, assumed to be identity)
- expresses that σ is a theory morphism $O_1 \to O_2$

```
logic CASL.FOL=
spec RichBooleanAlgebra =
  BooleanAlgebra
then %def
  pred __ <= __ : Elem * Elem;
  forall x,y:Elem
  . x <= y <=> x cap y = x %(leq_def)%
end
interpretation order_in_BA :
  PartialOrder to RichBooleanAlgebra
end
```

Recall Family Ontology

```
logic OWL
ontology Family2 =
  Class: Person
  Class: Woman SubClassOf: Person
  ObjectProperty: hasChild
  Class: Mother
         EquivalentTo: Woman and hasChild some Person
  Individual: mary Types: Woman Facts: hasChild  john
  Individual: john Types: Person
  Individual: mary
       Types: Annotations: Implied "true"^^xsd:boolean
              Mother
end
```

Interpretation in OWL

```
logic OWL
ontology Family_alt =
  Class: Human
  Class: Female
  Class: Woman EquivalentTo: Human and Female
  ObjectProperty: hasChild
  Class: Mother
        EquivalentTo: Female and hasChild some Human
end

interpretation i : Family_alt to Family2 =
  Human |-> Person, Female |-> Woman
end
```

Criterion for Theory Morphisms in Prop, FOL, OWL

Theorem

A signature morphism $\sigma : \Sigma_1 \rightarrow \Sigma_2$ *is a theory morphism* $\sigma : (\Sigma_1, \Gamma_1) \rightarrow (\Sigma_2, \Gamma_2)$ *iff*

$$\Gamma_2 \models_{\Sigma_2} \sigma(\Gamma_1)$$

Proof.

By the satisfaction condition. □

Implied extensions (in Prop, FOL, OWL)

The extension must not introduce new signature symbols:

$$(\Sigma, \Gamma)\ \textbf{then}\ (\emptyset, \Delta_\Gamma)$$

This leads to the theory morphism

$$(\Sigma, \Gamma) \xrightarrow{\iota} (\Sigma, \Gamma \cup \Delta_\Gamma)$$

The implied extension is well-formed if

$$\Gamma \models_\Sigma \Delta_\Gamma$$

That is, implied extensions are about logical consequence.

Conservative Extensions (in Prop, FOL, OWL)

Definition

A theory morphism $\sigma : T_1 \to T_2$ is consequence-theoretically conservative (ccons), if for each $\phi_1 \in \mathrm{Sen}(\Sigma_1)$

$$T_2 \models \sigma(\phi_1) \text{ implies } T_1 \models \phi_1.$$

(no “new” facts over the “old” signature)

Definition

A theory morphism $\sigma : T_1 \to T_2$ is model-theoretically conservative (mcons), if for each $M_1 \in \mathrm{Mod}(T_1)$, there is a σ-expansion

$$M_2 \in \mathrm{Mod}(T_2) \text{ with } (M_2)|_\sigma = M_1$$

A General Theorem

Theorem

In propositional logic, FOL and OWL, if $\sigma : T_1 \to T_2$ is mcons, then it is also ccons.

Proof.

Assume that $\sigma : T_1 \to T_2$ is mcons.
Let ϕ_1 be a formula, such that $T_2 \models_{\Sigma_2} \sigma(\phi_1)$.
Let M_1 be a model $M_1 \in Mod(T_1)$. By assumption there is a model $M_2 \in Mod(T_2)$ with $M_2|_\sigma = M_1$. Since $T_2 \models_{\Sigma_2} \sigma(\phi_1)$, we have $M_2 \models \sigma(\phi_1)$. By the satisfaction condition $M_2|_\sigma \models_{\Sigma_1} \phi_1$. Hence $M_1 \models \phi_1$. Altogether $T_1 \models_{\Sigma_1} \phi_1$. $\square$

Some prerequisites

Theorem (Compactness theorem for propositional logic)

If $\Gamma \models_\Sigma \phi$, *then* $\Gamma' \models_\Sigma \phi$ *for some finite* $\Gamma' \subseteq \Gamma$

Proof.

Logical consequence $\models_\Sigma$ *can be captured by provability* $\vdash_\Sigma$. *Proofs are finite.* □

Definition

Given a model $M \in \mathrm{Mod}(\Sigma)$, its theory $Th(M)$ is defined by

$$Th(M) = \{\varphi \in \mathrm{Sen}(\Sigma) \mid M \models_\Sigma \varphi\}$$

In Prop, the converse holds

Theorem

In propositional logic, if $\sigma : T_1 \to T_2$ is ccons, then it is also mcons.

Proof.

Assume that $\sigma : T_1 \to T_2$ *is ccons. Let* M_1 *be a model* $M_1 \in Mod(T_1)$. *Assume that* M_1 *has no* σ*-expansion to a* T_2*-model. This means that* $T_2 \cup \sigma(Th(M_1)) \models \bot$. *Hence by compactness we have* $T_2 \cup \sigma(\Gamma) \models \bot$ *for a finite* $\Gamma \subseteq Th(M_1)$. *Let* $\Gamma = \{\phi_1, \ldots, \phi_n\}$. *Thus* $T_2 \cup \sigma(\{\phi_1, \ldots, \phi_n\}) \models \bot$ *and hence* $T_2 \models \sigma(\phi_1) \wedge \ldots \wedge \sigma(\phi_n) \to \bot$. *This means* $T_2 \models \sigma(\phi_1 \wedge \ldots \wedge \phi_n \to \bot)$. *By assumption* $T_1 \models \phi_1 \wedge \ldots \wedge \phi_n \to \bot$. *Since* $M_1 \in Mod(T_1)$ *and* $M_1 \models \phi_i$ $(1 \leq i \leq n)$, *also* $M_1 \models \bot$. *Contradiction!* □

A Counterexample in ALC (ccons, not mcons)

```
logic OWL.ALC
ontology Service =
  ObjectProperty: provider
  ObjectProperty: input
  ObjectProperty: output
  Class: Webservice SubClassOf: provider some Thing
          and input some Thing and output some Thing
then %ccons
  Class: Array
  Class: Integer DisjointWith: Array
  Class: Webservice SubClassOf: input some Integer
         and input some Array
end
```

In OWL.SROIQ, this is not even ccons!

A Counterexample in FOL (ccons, not mcons)

```
logic CASL.FOL=
spec Weak_Nat =
  sort Nat ops 0:Nat succ: Nat -> Nat pred __<__ : Nat*Nat
  forall x,y,z : Nat
  . x = 0 \/ exists u:Nat . succ(u) = x
  . x < succ(y) <=> (x<y \/ x = y)
  . not (x < 0)
  . x < y => not (y < x)
  . (x < y /\ y < z) => (x < z)
  . x < y \/ x = y \/ y < x
then %ccons
  op __ + __ : Nat * Nat -> Nat
  forall x,y : Nat
  . 0 + y = y
  . succ(x) + y = succ(x + y) %(+succ)%
  . y < succ(x) + y           %(succ_great)%   end
```

Definitional Extensions (in Prop, FOL, OWL)

Definition

A theory morphism $\sigma : T_1 \to T_2$ is definitional, if for each $M_1 \in \mathrm{Mod}(T_1)$, there is a unique σ-expansion

$$M_2 \in \mathrm{Mod}(T_2) \text{ with } (M_2)|_\sigma = M_1$$

```
logic Propositional
spec Person =
  props person, male, female
then %def
  props man, woman
   . man <=> person /\ male
   . woman <=> person /\ female
end
```

Definitional Extensions: Example in OWL

```
logic OWL
ontology Person =
  Class: Person
  Class: Female
then %def
  Class: Woman  EquivalentTo: Person and Female
end
```

Summary of DOL Syntax for Extensions

- O_1 **then %mcons** O_2, O_1 **then %mcons** O_2: model-conservative extension
  - each O_1-model has an expansion to O_1 **then** O_2
- O_1 **then %ccons** O_2: consequence-conservative extension
  - O_1 **then** $O_2 \models \varphi$ implies $O_1 \models \varphi$, for φ in the language of O_1
- O_1 **then %def** O_2: definitional extension
  - each O_1-model has a unique expansion to O_1 **then** O_2
- O_1 **then %implies** O_2: implied extension
  - like %mcons, but O_2 must not extend the signature

Scaling it to the Web

- OMS can be referenced directly by their URL (or IRI)

```
<http://owl.cs.manchester.ac.uk/co-ode-files/ontologies/
pizza.owl>
```

- Prefixing may be used for abbreviation

```
%prefix( co-ode:
  <http://owl.cs.manchester.ac.uk/co-ode-files/ontologies/>
       )%
co-ode:pizza.owl
```

Exercise for tomorrow

- if you not have done so already, clone the ESSLLI repository on `ontohub.org`:
  `git clone git://ontohub.org/esslli-2016.git`

Exercise for tomorrow

- if you not have done so already, clone the ESSLLI repository on `ontohub.org`:
  `git clone git://ontohub.org/esslli-2016.git`
- Look at the theories

Exercise for tomorrow

- if you not have done so already, clone the ESSLLI repository on `ontohub.org`:
  `git clone git://ontohub.org/esslli-2016.git`
- Look at the theories
- (Dis)prove theorems (both with Hets and on Ontohub.org)

Exercise for tomorrow

- if you not have done so already, clone the ESSLLI repository on `ontohub.org`:
  `git clone git://ontohub.org/esslli-2016.git`
- Look at the theories
- (Dis)prove theorems (both with Hets and on Ontohub.org)
- Write some theory on your own, add intended consequences and prove them