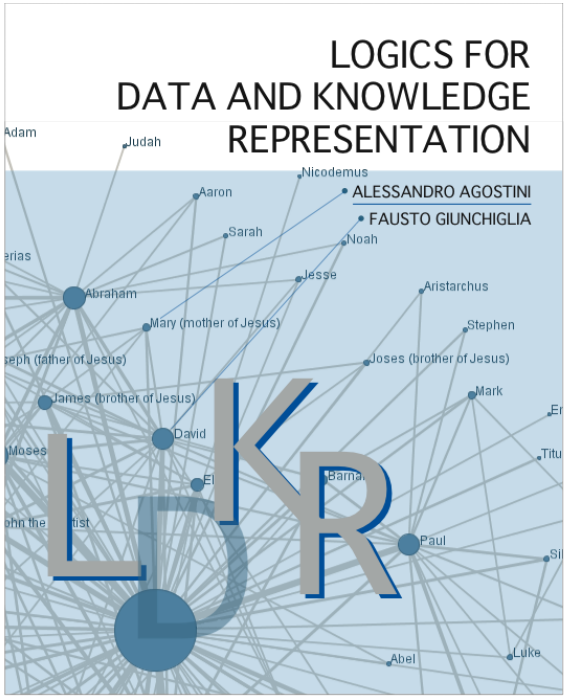


Logics for Data and Knowledge Representation

Exercises: ClassL

Fausto Giunchiglia, Mattia Fumagalli, Rui Zhang ,Vincenzo Maltese

Outline

- Syntax
  - Symbols and formation rules
- Reasoning with a TBox
- Reasoning with an ABox

Symbols in ClassL

1. Which of the following symbols are used in ClassL?

$$\sqcap \quad \neg \quad \top \quad \vee \quad \equiv \quad \sqcup \quad \sqsubseteq \quad \longrightarrow \quad \leftrightarrow \quad \perp \quad \wedge \quad \models$$

2. Which of the following symbols are in well formed formulas?

$$\sqcap \quad \neg \quad \top \quad \vee \quad \equiv \quad \sqcup \quad \sqsubseteq \quad \longrightarrow \quad \leftrightarrow \quad \perp \quad \wedge \quad \models$$

Symbols in ClassL (solution)

1. Which of the following symbols are used in ClassL?

$\sqcap$ $\neg$ $\top$ $\vee$ $\equiv$ $\sqcup$ $\sqsubseteq$ $\rightarrow$ $\leftrightarrow$ $\bot$ $\wedge$ $\models$

2. Which of the following symbols are in well formed formulas?

$\sqcap$ $\neg$ $\top$ $\vee$ $\equiv$ $\sqcup$ $\sqsubseteq$ $\rightarrow$ $\leftrightarrow$ $\bot$ $\wedge$ $\models$

Remember the BNF grammar: <Atomic Formula> ::= A | B | ... | P | Q | ... | $\bot$ | $\top$

<wff> ::= <Atomic Formula> | ¬<wff> | <wff> $\sqcap$ <wff> | <wff> $\sqcup$ <wff> |

<Atomic Formula> $\sqsubseteq$ <wff> | <Atomic Formula> ≡ <wff>

Formation rules

- Which of the following is not a wff in ClassL?

1. **¬ MonkeyLow ⊔ BananaHigh**
2. **¬ ¬ MonkeyLow ⊓ BananaHigh ⊑ ¬ GetBanana**
3. **MonkeyLow ¬ ⊓ BananaHigh**
4. **MonkeyLow ∨ ¬ GetBanana**

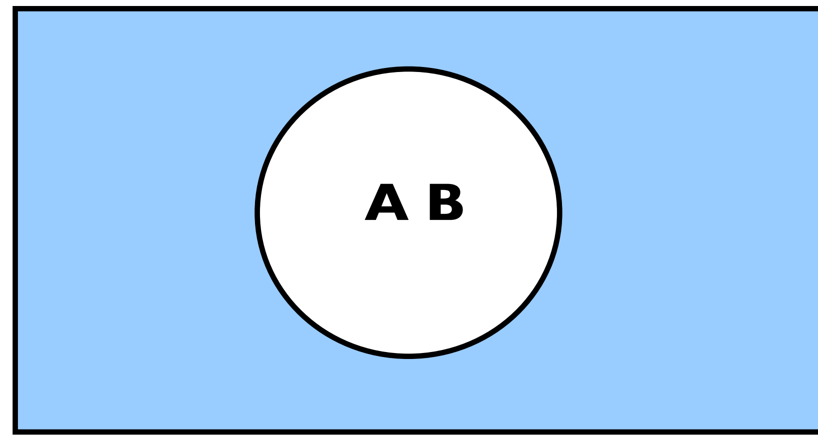
NUM 2, 3, 4 !

Satisfiability

- Given the TBox T={$A \sqsubseteq B$, $B \sqsubseteq A$}, is $\neg(A \sqcap B)$ satisfiable in ClassL?

RECALL: to prove satisfiability it is enough to find <u>one</u> model.

We can use Venn Diagrams.



In alternative, this can be proved with entailment + DPLL:

P: $A \sqsubseteq B$ Q: $B \sqsubseteq A$ S: $\neg(A \sqcap B)$

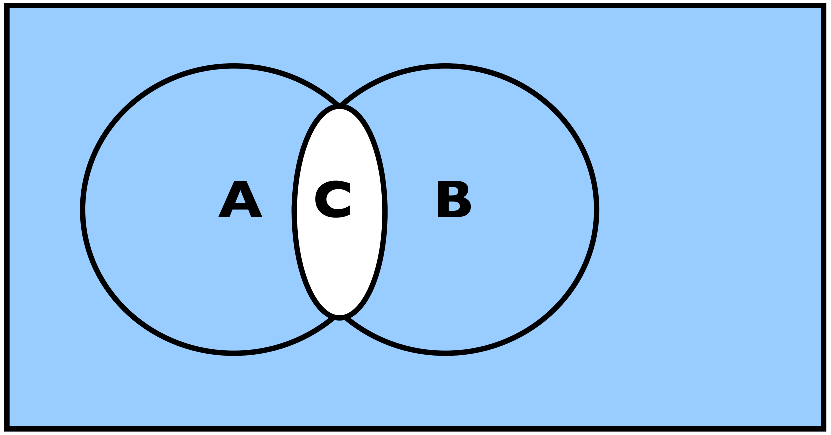
$\{P, Q\} \models S$

DPLL ($\neg$(RewriteInPL(P) $\wedge$ RewriteInPL(Q) $\rightarrow$ RewriteInPL(S))

DPLL($\neg((A \rightarrow B) \wedge (B \rightarrow A)) \rightarrow \neg(A \wedge B)$)

ClassL Exercises 2

- Given the TBox T={C⊑A, C⊑B} is ¬(A⊓B) satisfiable in ClassL?



Satisfiability with respect to a TBox T

RECALL:

A concept P is satisfiable w.r.t. a terminology T, if there exists an interpretation I with I $\models$ θ for all $\theta \in$ T, and such that I $\models$ P, namely I(P) is not empty

Satisfiability with respect to a TBox T

- Suppose we model the Monkey-Banana problem as follows:
  "*If the monkey is low in position then it cannot get the banana. If the monkey gets the banana it survives*".

  TBox T

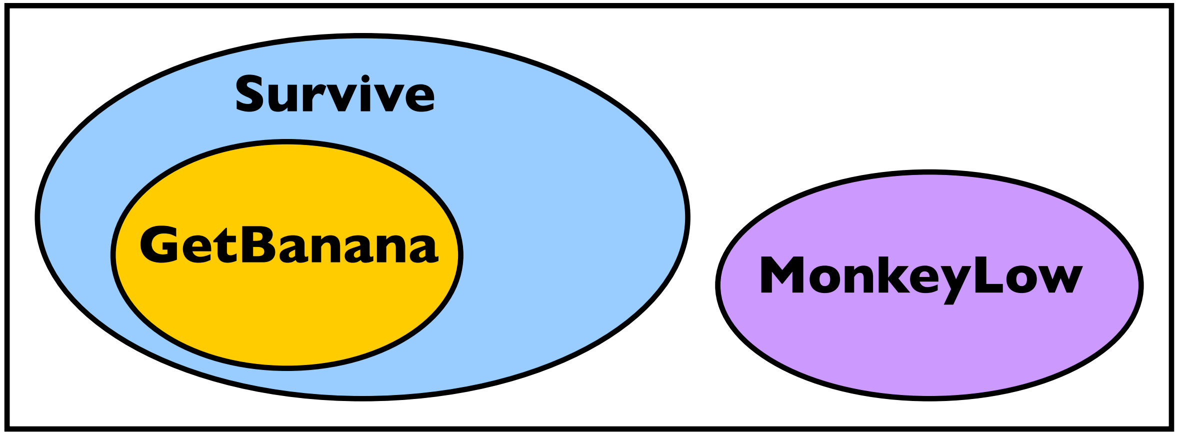
  MonkeyLow $\sqsubseteq$ $\neg$ GetBanana

  GetBanana $\sqsubseteq$ Survive

- Is T satisfiable?

  YES! Look at the

  Venn diagram

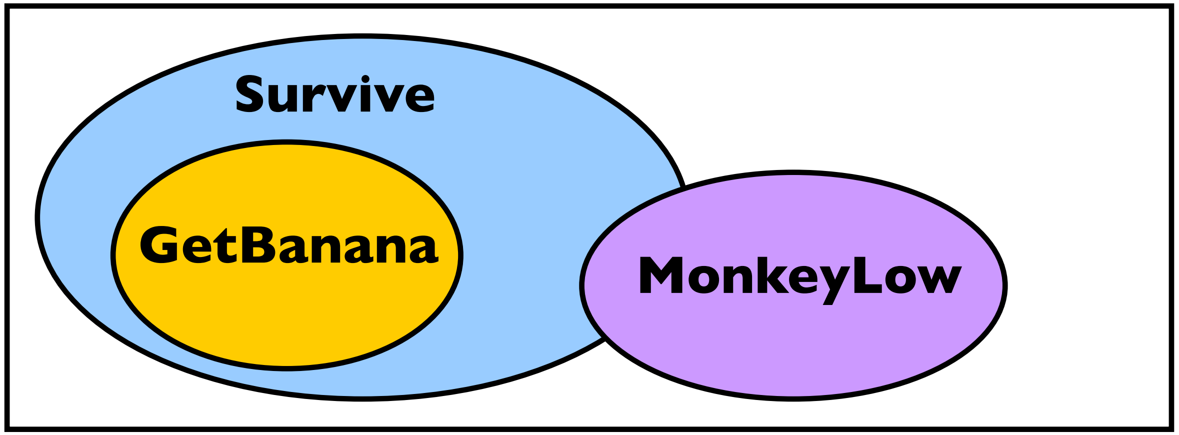


Satisfiability with respect to a TBox T

- Suppose we model the Monkey-Banana problem as follows:
  TBox T
  MonkeyLow $\sqsubseteq$ $\neg$ GetBanana
  GetBanana $\sqsubseteq$ Survive

- Is it possible for a monkey to survive even if it does not get the banana?
- We can restate the problem as follow:
  does T $\models$ $\neg$ GetBanana $\sqcap$ Survive ?

YES! Look at the Venn diagram



Normalization of a TBox

- Normalize the TBox below:

  $\text{MonkeyLow} \sqsubseteq \neg\, \text{GetBanana}$

  $\text{GetBanana} \equiv \text{Survive}$

- Possible solution:

  $\text{MonkeyLow} \equiv \neg\, \text{GetBanana} \sqcap \neg\, \text{ClimbBox}$

  $\text{GetBanana} \equiv \text{Survive}$

  Note that, with this theory, the monkey necessarily needs to get the banana to survive.

Expansion of a TBox

- Expand the TBox below:

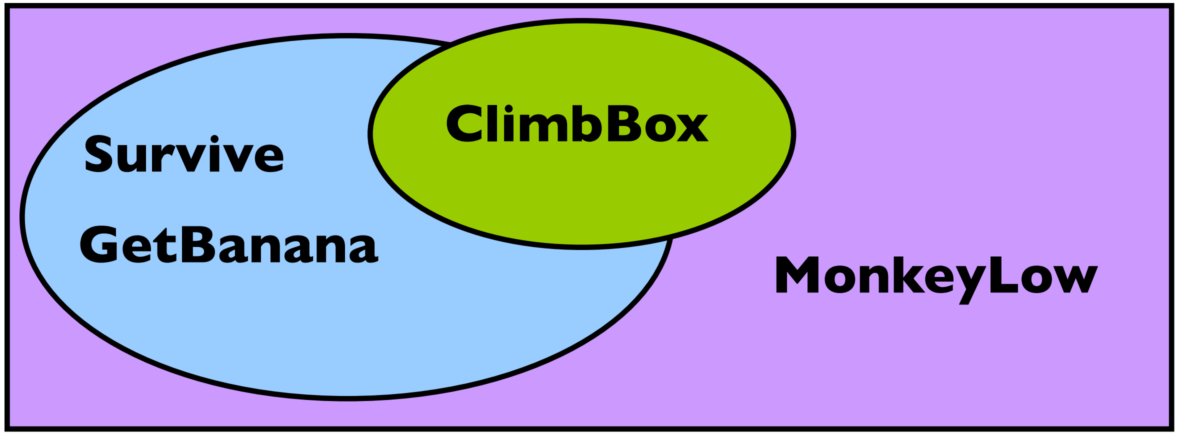
  MonkeyLow ≡ ¬ GetBanana ⊓ ¬ ClimbBox

  GetBanana ≡ Survive

- T', expansion of T (The Venn diagram gives a possible model):

  MonkeyLow ≡ ¬ Survive ⊓ ¬ ClimbBox

  GetBanana ≡ Survive



Notice that the fact that a monkey climbs the box does not necessarily mean that it survives.

ABox: Consistency

- Check the consistency of A w.r.t. T via expansion.

T	A
MonkeyLow ≡ ¬ GetBanana ⊓ ¬ ClimbBox	MonkeyLow(Cita)
GetBanana ≡ Survive	¬Survive(Cita)

Expansion of A is consistent:

MonkeyLow(Cita)

¬ GetBanana(Cita)

¬ ClimbBox(Cita)

¬ Survive(Cita)

¬ GetBanana(Cita)

ABox: Instance checking

- Given T and A below

T	A
MonkeyLow ≡ ¬ GetBanana ⊓ ¬ ClimbBox GetBanana ≡ Survive	MonkeyLow(Cita) ¬Survive(Cita)

- Is Cita an instance of MonkeyLow?

  YES

- Is Cita an instance of ClimbBox?

  NO

- Is Cita an instance of GetBanana?

  NO

Expansion of A
MonkeyLow(Cita)
¬ GetBanana(Cita)
¬ClimbBox(Cita)

¬ Survive(Cita)
¬GetBanana(Cita)

Instance Retrieval. Consider the following expansion…

T

Undergraduate	$\sqsubseteq \neg$ Teach
Bachelor	$\equiv$ Student $\sqcap$ Undergraduate
Master	$\equiv$ Student $\sqcap \neg$ Undergraduate
PhD	$\equiv$ Master $\sqcap$ Research
Assistant	$\equiv$ PhD $\sqcap$ Teach

A

Master(Chen)
PhD(Enzo)
Assistant(Rui)

The expansion of A

Master(Chen)	PhD(Enzo)	Assistant(Rui)
Student(Chen)	Master(Enzo)	PhD(Rui)
¬Undergraduate(Chen)	Research(Enzo)	Teach(Rui)
	Student(Enzo)	Master(Rui)
	¬Undergraduate(Enzo)	Research(Rui)
		Student(Rui)
		¬Undergraduate(Rui)

Instance Retrieval. … find the instances of Master

T

Undergraduate $\sqsubseteq \neg$ Teach
Bachelor $\equiv$ Student $\sqcap$ Undergraduate
Master $\equiv$ Student $\sqcap \neg$ Undergraduate
PhD $\equiv$ Master $\sqcap$ Research
Assistant $\equiv$ PhD $\sqcap$ Teach

A

Master(Chen)
PhD(Enzo)
Assistant(Rui)

The expansion of A

Master(Chen)	PhD(Enzo)	Assistant(Rui)
Student(Chen)	**Master(Enzo)**	PhD(Rui)
¬Undergraduate(Chen)	Research(Enzo)	Teach(Rui)
	Student(Enzo)	**Master(Rui)**
	¬Undergraduate(Enzo)	Research(Rui)
		Student(Rui)
		¬Undergraduate(Rui)

ABox: Concept realization

- Find the most specific concept C such that A ⊨ C(Cita)

T
MonkeyLow ≡ ¬ GetBanana ⊓ ¬ ClimbBox
GetBanana ≡ Survive

A
MonkeyLow(Cita)
¬Survive(Cita)

Notice that MonkeyLow directly uses GetBanana and ClimbBox, and it uses Survive. The most specific concept is therefore MonkeyLow.

Defining the TBox and ABox: the LDKR Class

- Define a TBox and ABox for the following database:

LDKR

Name	Nationality	Hair
Fausto	Italian	White
Enzo	Italian	Black
Rui	Chinese	Black
Bisu	Indian	Black

NOTE: ClassL is not expressive enough to represent database constrains such as keys involving two fields.

ABox =
{Italian(Fausto), Italian(Enzo), Chinese(Rui), Indian(Bisu), BlackHair(Enzo), BlackHair(Rui), BlackHair(Bisu), WhiteHair(Fausto)}

TBox =
{Italian $\sqsubseteq$ LDKR,
Indian $\sqsubseteq$ LDKR,
Chinese $\sqsubseteq$ LDKR,
BlackHair $\sqsubseteq$ LDKR,
WhiteHair $\sqsubseteq$ LDKR}

ALC and more complex DLs

Exercise

Represent Metro lines in Milan in a labelled directed graph

ALC and more complex DLs

Exercise

Define Σ for speaking about the metro in Milan, and give examples of Concepts, Definitions, Subsumptions, and Assertions

Solution (Syntax)

- *Concept Names* (Σ_C):

Station	*the set of metro stations*
RedLineStation	*the set of metro stations on the red line*
ExchangeStation	*the set of metro stations where to change line*

- *Role Names* (Σ_R):

Next	*the relation between one station and its next stations*

- *Individual Names* (Σ_I):

Centrale	*the station called "Centrale"*
Gioia	*the station called "Gioia" . . .*

ALC and more complex DLs

Solution (Concepts)

the set of stations which are on both the red and green line

$RedLineStation \sqcap GreenLineStation$

the set of exchange stations on the red line

$ExchangeStation \sqcap RedLineStation$

the set of stations which have a next station on the red line

$Station \sqcap \exists Next.RedLineStation$

The set of End stations

$Station \sqcap \forall Next.\bot$

ALC and more complex DLs

Solution (Definitions)

$$
\begin{aligned}
RGExchangeStation &\doteq RedLineStation \sqcap GreenLineStation \\
RYExchangeStation &\doteq RedLineStation \sqcap YellowLineStation \\
GYExchangeStation &\doteq GreenLineStation \sqcap YellowLineStation \\
ExchangeStation &\doteq RGExchangeStation \sqcup RYExchangeStation \\
&\quad \sqcup GYExchangeStation
\end{aligned}
$$

ALC and more complex DLs

Solution (Subsumptions)

A red line station is a station

$RedLineStation \sqsubseteq Station$

everything next to something is a station

$\top \sqsubseteq \forall Next.Station$

everything that has something next must be a station

$\exists Next.\top \sqsubseteq Station$

ALC and more complex DLs

Solution (Subsumptions)

A red line station is a station

$RedLineStation \sqsubseteq Station$

everything next to something is a station

$\top \sqsubseteq \forall Next.Station$

everything that has something next must be a station

$\exists Next.\top \sqsubseteq Station$

ALC and more complex DLs

Solution (Assertions)

"Gioia" is a station of the green line

$GreenLineStation(Gioia)$

"Loreto" is an exchange station between the green and the red line

$RGExchangeStation(Loreto)$

"Lima" is the stop that follows "Loreto"

$Next(Loreto, Lima)$

"Duomo" is not the next stop of "Loreto"

$\neg Next(Loreto, Duomo)$