

- A proposition is a sentence, written in a language, that has a truth value – it is true or false – in a world. A proposition is built from atomic propositions (atoms) and logical connectives.

Propositions

- A proposition is a sentence, written in a language, that has a truth value – it is true or false – in a world. A proposition is built from atomic propositions (atoms) and logical connectives.
- Propositions can be built from simpler propositions using logical connectives.

Propositional Calculus Syntax

- An **atomic proposition** – **atom** – is a symbol, written as sequences of letters, digits, and the underscore (`_`) and start with a lower-case letter.

E.g., `a`, `ai_is_fun`, `lit_I1`, `live_outside`, `mimsy`, `sunny`.

Propositional Calculus Syntax

- An **atomic proposition** – **atom** – is a symbol, written as sequences of letters, digits, and the underscore (`_`) and start with a lower-case letter.

E.g., `a`, `ai_is_fun`, `lit_1`, `live_outside`, `mimsy`, `sunny`.

- A **proposition** or **logical formula** is either
 - ▶ an atomic proposition or
 - ▶ a **compound proposition** of the form

$\neg p$	“not p ”	negation of p
$p \wedge q$	“ p and q ”	conjunction of p and q
$p \vee q$	“ p or q ”	disjunction of p and q
$p \rightarrow q$	“ p implies q ”	implication of q from p
$p \leftarrow q$	“ p if q ”	implication of p from q
$p \leftrightarrow q$	“ p if and only if q ”	equivalence of p and q
$p \oplus q$	“ p XOR q ”	exclusive-or of p and q

where p and q are propositions.

Propositional Calculus Syntax

- An **atomic proposition** – **atom** – is a symbol, written as sequences of letters, digits, and the underscore (`_`) and start with a lower-case letter.

E.g., `a`, `ai_is_fun`, `lit_1`, `live_outside`, `mimsy`, `sunny`.

- A **proposition** or **logical formula** is either
 - ▶ an atomic proposition or
 - ▶ a **compound proposition** of the form

$\neg p$	“not p ”	negation of p
$p \wedge q$	“ p and q ”	conjunction of p and q
$p \vee q$	“ p or q ”	disjunction of p and q
$p \rightarrow q$	“ p implies q ”	implication of q from p
$p \leftarrow q$	“ p if q ”	implication of p from q
$p \leftrightarrow q$	“ p if and only if q ”	equivalence of p and q
$p \oplus q$	“ p XOR q ”	exclusive-or of p and q

where p and q are propositions.

- The operators \neg , \wedge , \vee , \rightarrow , \leftarrow , \leftrightarrow , and \oplus are **logical connectives**.

Why propositions?

- Logical formulae are modular statements of what is (known to be) true

Why propositions?

- Logical formulae are modular statements of what is (known to be) true
- It is easier to check correctness and debug formulae than tables of what could be true

Why propositions?

- Logical formulae are modular statements of what is (known to be) true
- It is easier to check correctness and debug formulae than tables of what could be true
- We can exploit the Boolean nature for efficient reasoning

Why propositions?

- Logical formulae are modular statements of what is (known to be) true
- It is easier to check correctness and debug formulae than tables of what could be true
- We can exploit the Boolean nature for efficient reasoning
- We need a language for asking queries (of what follows in all models) that may be more complicated than asking for the value of a variable

Why propositions?

- Logical formulae are modular statements of what is (known to be) true
- It is easier to check correctness and debug formulae than tables of what could be true
- We can exploit the Boolean nature for efficient reasoning
- We need a language for asking queries (of what follows in all models) that may be more complicated than asking for the value of a variable
- It is easy to incrementally add formulae

Why propositions?

- Logical formulae are modular statements of what is (known to be) true
- It is easier to check correctness and debug formulae than tables of what could be true
- We can exploit the Boolean nature for efficient reasoning
- We need a language for asking queries (of what follows in all models) that may be more complicated than asking for the value of a variable
- It is easy to incrementally add formulae
- It can be extended to infinitely many propositions with infinite domains (using logical quantification)

Semantics of the Propositional Calculus

- An **interpretation** – or **possible world** – is an assignment of true or false to each variable.

Semantics of the Propositional Calculus

- An **interpretation** – or **possible world** – is an assignment of true or false to each variable.
- An interpretation is defined by function π that maps **atoms** to $\{true, false\}$.
If $\pi(a)=true$, atom a is **true** in the interpretation.
If $\pi(a)=false$, atom a is **false** in the interpretation.

Semantics of the Propositional Calculus

- An **interpretation** – or **possible world** – is an assignment of true or false to each variable.
- An interpretation is defined by function π that maps **atoms** to $\{true, false\}$.
If $\pi(a)=true$, atom a is **true** in the interpretation.
If $\pi(a)=false$, atom a is **false** in the interpretation.
- Truth of a compound proposition in an interpretation is defined in terms of the truth of its components:

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftarrow q$	$p \leftrightarrow q$	$p \oplus q$
<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>
<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>
<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>

- Propositions can have different truth values in different interpretations.

Models and Logical Consequence

- A **model** of a set of clauses is an interpretation in which all the clauses are *true*.
- If KB is a set of propositions, proposition g is a **logical consequence** of KB , written $KB \models g$, if g is *true* in every model of KB .
- That is, $KB \models g$ if there is no interpretation in which KB is *true* and g is *false*.

Simple Example

$$KB = \begin{cases} \text{apple_eaten} \leftarrow \text{bird_eats_apple.} \\ \text{light_on} \leftarrow \text{night.} \\ \text{night.} \end{cases}$$

	<i>apple_eaten</i>	<i>bird_eats_apple</i>	<i>light_on</i>	<i>night</i>	model of <i>KB</i> ?
<i>l</i> ₁	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	
<i>l</i> ₂	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	
<i>l</i> ₃	<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>	
<i>l</i> ₄	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	
<i>l</i> ₅	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	

Simple Example

$$KB = \begin{cases} \text{apple_eaten} \leftarrow \text{bird_eats_apple.} \\ \text{light_on} \leftarrow \text{night.} \\ \text{night.} \end{cases}$$

	<i>apple_eaten</i>	<i>bird_eats_apple</i>	<i>light_on</i>	<i>night</i>	model of <i>KB</i> ?
<i>l</i> ₁	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	yes
<i>l</i> ₂	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	
<i>l</i> ₃	<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>	
<i>l</i> ₄	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	
<i>l</i> ₅	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	

Simple Example

$$KB = \begin{cases} \text{apple_eaten} \leftarrow \text{bird_eats_apple}. \\ \text{light_on} \leftarrow \text{night}. \\ \text{night}. \end{cases}$$

	<i>apple_eaten</i>	<i>bird_eats_apple</i>	<i>light_on</i>	<i>night</i>	model of <i>KB</i> ?
l_1	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	yes
l_2	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	no
l_3	<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>	
l_4	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	
l_5	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	

Simple Example

$$KB = \begin{cases} \text{apple_eaten} \leftarrow \text{bird_eats_apple}. \\ \text{light_on} \leftarrow \text{night}. \\ \text{night}. \end{cases}$$

	<i>apple_eaten</i>	<i>bird_eats_apple</i>	<i>light_on</i>	<i>night</i>	model of <i>KB</i> ?
<i>l</i> ₁	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	yes
<i>l</i> ₂	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	no
<i>l</i> ₃	<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>	no
<i>l</i> ₄	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	
<i>l</i> ₅	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	

Simple Example

$$KB = \begin{cases} \text{apple_eaten} \leftarrow \text{bird_eats_apple.} \\ \text{light_on} \leftarrow \text{night.} \\ \text{night.} \end{cases}$$

	<i>apple_eaten</i>	<i>bird_eats_apple</i>	<i>light_on</i>	<i>night</i>	model of <i>KB</i> ?
<i>l</i> ₁	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	yes
<i>l</i> ₂	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	no
<i>l</i> ₃	<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>	no
<i>l</i> ₄	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	yes
<i>l</i> ₅	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	

Simple Example

$$KB = \begin{cases} \text{apple_eaten} \leftarrow \text{bird_eats_apple.} \\ \text{light_on} \leftarrow \text{night.} \\ \text{night.} \end{cases}$$

	<i>apple_eaten</i>	<i>bird_eats_apple</i>	<i>light_on</i>	<i>night</i>	model of <i>KB</i> ?
<i>l</i> ₁	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	yes
<i>l</i> ₂	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	no
<i>l</i> ₃	<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>	no
<i>l</i> ₄	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	yes
<i>l</i> ₅	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	yes

Simple Example

$$KB = \begin{cases} \text{apple_eaten} \leftarrow \text{bird_eats_apple.} \\ \text{light_on} \leftarrow \text{night.} \\ \text{night.} \end{cases}$$

	<i>apple_eaten</i>	<i>bird_eats_apple</i>	<i>light_on</i>	<i>night</i>	model of <i>KB</i> ?
<i>l</i> ₁	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	yes
<i>l</i> ₂	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	no
<i>l</i> ₃	<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>	no
<i>l</i> ₄	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	yes
<i>l</i> ₅	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	yes

Which of *apple_eaten*, *bird_eats_apple*, *light_on*, *night* logically follow from *KB*?

Simple Example

$$KB = \begin{cases} \text{apple_eaten} \leftarrow \text{bird_eats_apple}. \\ \text{light_on} \leftarrow \text{night}. \\ \text{night}. \end{cases}$$

	<i>apple_eaten</i>	<i>bird_eats_apple</i>	<i>light_on</i>	<i>night</i>	model of <i>KB</i> ?
<i>l</i> ₁	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	yes
<i>l</i> ₂	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	no
<i>l</i> ₃	<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>	no
<i>l</i> ₄	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	yes
<i>l</i> ₅	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	yes

Which of *apple_eaten*, *bird_eats_apple*, *light_on*, *night* logically follow from *KB*?

$KB \models \text{light_on}$, $KB \models \text{night}$,

$KB \not\models \text{apple_eaten}$, $KB \not\models \text{bird_eats_apple}$

Step 1 Begin with a task domain.

Human's view of semantics

Step 1 Begin with a task domain.

Step 2 Choose atoms in the computer to denote propositions. These atoms have meaning to the KB designer.

Human's view of semantics

- Step 1 Begin with a task domain.
- Step 2 Choose atoms in the computer to denote propositions. These atoms have meaning to the KB designer.
- Step 3 Tell the system knowledge about the domain.

Human's view of semantics

- Step 1 Begin with a task domain.
- Step 2 Choose atoms in the computer to denote propositions. These atoms have meaning to the KB designer.
- Step 3 Tell the system knowledge about the domain.
- Step 4 Ask the system questions.

Human's view of semantics

Step 1 Begin with a task domain.

Step 2 Choose atoms in the computer to denote propositions. These atoms have meaning to the KB designer.

Step 3 Tell the system knowledge about the domain.

Step 4 Ask the system questions.

— The system can tell you whether the question is a logical consequence.

Human's view of semantics

Step 1 Begin with a task domain.

Step 2 Choose atoms in the computer to denote propositions. These atoms have meaning to the KB designer.

Step 3 Tell the system knowledge about the domain.

Step 4 Ask the system questions.

— The system can tell you whether the question is a logical consequence.

— You can interpret the answer with the meaning associated with the atoms.

- The computer doesn't have access to the intended interpretation.

Computer's view of semantics

- The computer doesn't have access to the intended interpretation.
- All it knows is the knowledge base.

Computer's view of semantics

- The computer doesn't have access to the intended interpretation.
- All it knows is the knowledge base.
- The computer can determine if a formula is a logical consequence of KB.

Computer's view of semantics

- The computer doesn't have access to the intended interpretation.
- All it knows is the knowledge base.
- The computer can determine if a formula is a logical consequence of KB.
- If $KB \models g$ then g

Computer's view of semantics

- The computer doesn't have access to the intended interpretation.
- All it knows is the knowledge base.
- The computer can determine if a formula is a logical consequence of KB.
- If $KB \models g$ then g must be true in the intended interpretation.

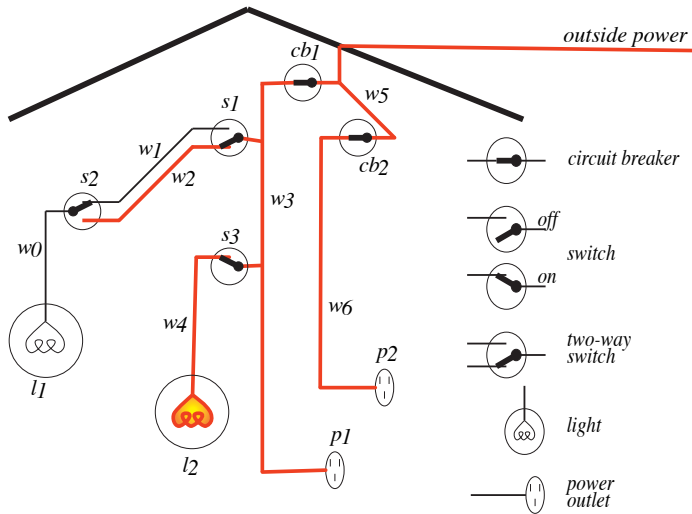
Computer's view of semantics

- The computer doesn't have access to the intended interpretation.
- All it knows is the knowledge base.
- The computer can determine if a formula is a logical consequence of KB.
- If $KB \models g$ then g must be true in the intended interpretation.
- If $KB \not\models g$ then

Computer's view of semantics

- The computer doesn't have access to the intended interpretation.
- All it knows is the knowledge base.
- The computer can determine if a formula is a logical consequence of KB.
- If $KB \models g$ then g must be true in the intended interpretation.
- If $KB \not\models g$ then there is a model of KB in which g is false. This could be the intended interpretation.

Electrical Environment



In computer:

$light2_broken \leftarrow power_in_w_3$
 $\wedge sw_3_up \wedge unlit_light2.$

$sw_3_up.$

$power_in_w_3 \leftarrow power_in_p_1.$

$unlit_light2.$

$power_in_p_1.$

In user's mind:

- $light2_broken$: light #2 is broken
- sw_3_up : switch 3 is up
- $power_in_w_3$: there is power in wire 3
- $unlit_light2$: light #2 isn't lit
- $power_in_p_1$: outlet p_1 has power

Conclusion:

In computer:

$light2_broken \leftarrow power_in_w_3$
 $\wedge sw_3_up \wedge unlit_light2.$

$sw_3_up.$

$power_in_w_3 \leftarrow power_in_p_1.$

$unlit_light2.$

$power_in_p_1.$

In user's mind:

- $light2_broken$: light #2 is broken
- sw_3_up : switch 3 is up
- $power_in_w_3$: there is power in wire 3
- $unlit_light2$: light #2 isn't lit
- $power_in_p_1$: outlet p_1 has power

Conclusion: $light2_broken$

In computer:

$light2_broken \leftarrow power_in_w_3$
 $\wedge sw_3_up \wedge unlit_light2.$

$sw_3_up.$

$power_in_w_3 \leftarrow power_in_p_1.$

$unlit_light2.$

$power_in_p_1.$

In user's mind:

- $light2_broken$: light #2 is broken
- sw_3_up : switch 3 is up
- $power_in_w_3$: there is power in wire 3
- $unlit_light2$: light #2 isn't lit
- $power_in_p_1$: outlet p_1 has power

Conclusion: $light2_broken$

- The computer doesn't know the meaning of the symbols
- The user can interpret symbols using their meaning