Logic and Computer Science

Why should computer scientists learn about logic?

1. Modelling of tasks for computer automation:
   - Computers are often used to automate processes previously performed by people.
   - This automation requires precise modelling of processes which are handled by humans in completely different ways.

2. Specific applications:
   - Description of programming language semantics
   - Correctness of computer systems and protocols
   - Description of computer hardware (“logic design”)
   - Automated reasoning
   - Modelling for intelligent systems

The Idea of a Logic

Two fundamental components:

- **Semantics**: A collection of *configurations* or *possible worlds* to be modelled. In formal mathematics, these are sometimes called *structures* or *interpretations.* The number of configurations is often extremely large, sometimes infinite, and so it is impossible to describe them and reason about them directly.

- **Syntax**: A language for describing properties of *possible worlds*, together with a set of rules for reasoning about them.

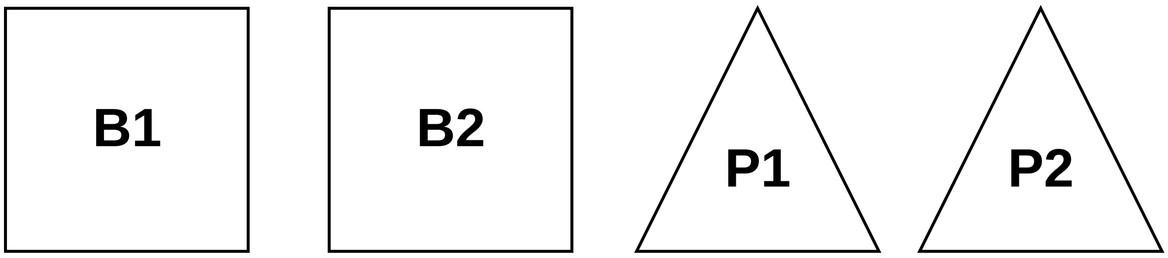
One cannot understand logic unless one understands these two ideas, and the relationship between them. A logic consists of both components, *together with precise rules for relating one to the other.*

Despite this fact, many (if not most) books which deal with logic in an applied manner (*e.g.,* books on artificial intelligence) fail to make this distinction clear.

We therefore begin with a simple example, designed to illustrate these two notions.

A Blocks-World Example

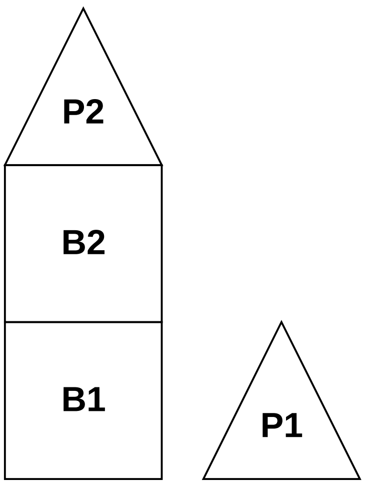
- Two cubes and two pyramids sitting on a table.



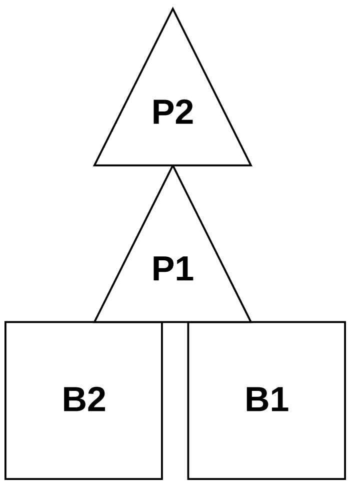
Assumptions:

- Cubes may be stacked, and pyramids may be placed atop cubes.
- Only one-on-one stacking is allowed.

Allowed:

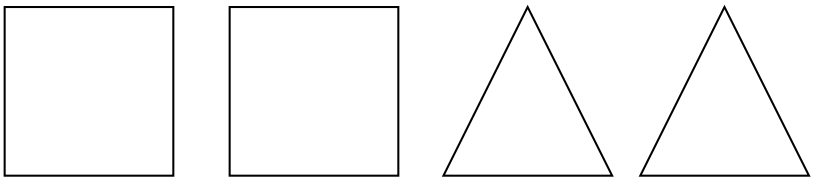


Not allowed:

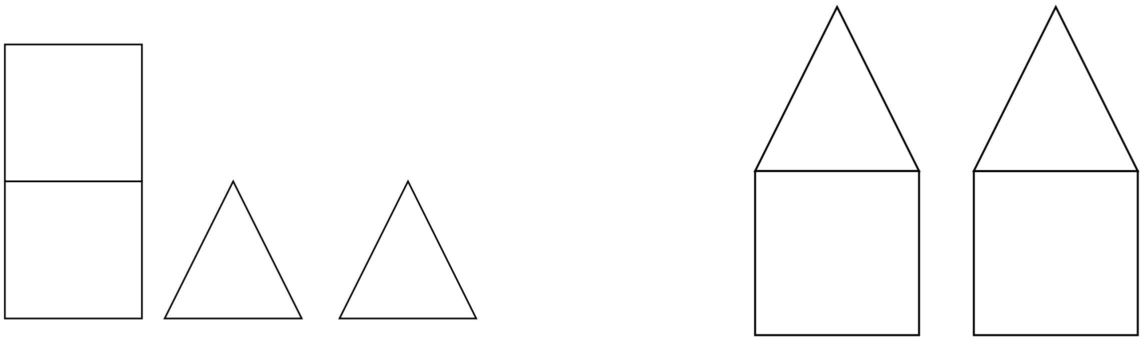


There are 13 possible worlds in this simple block setting.

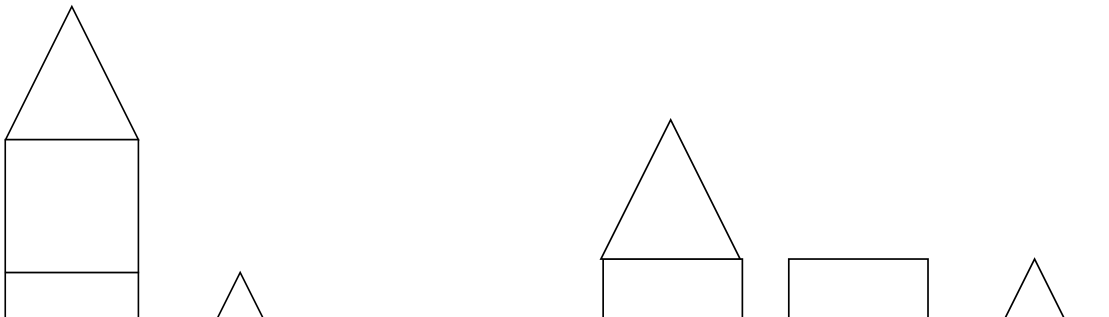
One with this arrangement:



Two variations with each of these two arrangements:



Four variations with each of these two arrangements:



A syntax for the blocks world:

In the table below, x, y ∈ {B1, B2, P1, P2}.

Proposition Schema	Meaning
On_table(x)	Object x is on the table.
On(x,y)	Object x is atop object y.
Is_cube(x)	Object x is a cube.
Is_pyramid(x)	Object x is a pyramid.

Examples: On_table(B1), On(P1,B1), Is_cube(B1), Is_Pyramid(B2).

- Some propositions are true in all possible worlds.
  Example: Is_cube(B1), Is_Pyramid(P1).

- Some propositions are false in all possible worlds.
  Examples: Is_cube(P1), On(B1,P1).

- Some propositions are true in some worlds and false in others.
  Examples: On(P1,B1), On_table(P2).

A possible world in which a proposition P is true is called a *model* of P.

The set of all models of P is denoted Mod(P).

Exercise: How many models of On(P1,B1) are there? List them.

Logical connectives may be used to combine propositions. The common logical connectives are identified in the following table:

Symbol	Arity	Precedence	Meaning
$\wedge$	2	2	Logical conjunction (and)
$\vee$	2	3	Logical disjunction (or)
$\neg$	1	1	Logical negation (not)
$\rightarrow$	2	4	Logical implication (implies)
$\leftrightarrow$ $\equiv$	2	4	Logical equivalence (iff)

- $\wedge$ and $\vee$ are commutative and associative.
- $\neg$ is its own inverse.
- Usually, it is safest to assume that all binary operators have the same precedence, and to use parentheses to avoid any ambiguity.
- With logical connectives, propositions may be combined to form well-formed formulas (wff's).

Examples of wff's which are true in all possible worlds:

- Is_cube(B1)
- Is_cube(B1) $\vee$ On_table(P1)
- Is_cube(B1) $\vee$ On(B2,P2)
- $\neg$Is_pyramid(B1)
- (On_table(B1) $\vee$ $\neg$On_table(B1))

- $(\text{On\_table}(B1) \vee \text{On\_table}(B2))$
- $(\text{On}(B1,B2) \rightarrow (\text{On\_table}(P1) \vee \text{On\_table}(P2))$

Wff's which are false in all possible worlds:

- Is_Cube(P1)
- $\neg$Is_Cube(B1)
- $(\text{On\_table}(B1) \wedge \neg\text{On\_table}(B1))$
- On(B1,P1)
- $\text{On}(B1,B2) \wedge \text{On}(P1,B2)$

Wff's which are true in some worlds and false in others:

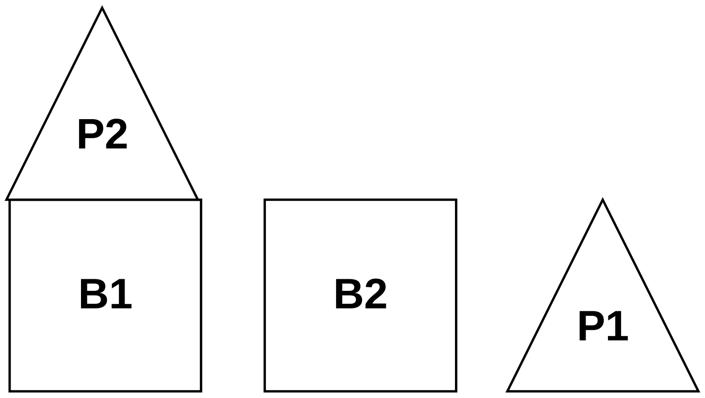
- $\text{On\_table}(B1) \wedge \text{On\_table}(P1)$
- $\text{On\_table}(B1) \rightarrow (\text{On\_table}(B2) \vee \text{On}(P1,B1))$

Note: The concept of Mod extends to wff's in an obvious fashion.

Describing a single possible world:

In the blocks-world example, it is possible to obtain a description of a given world by listing all of the propositions which are true for that world.

Example:



The propositions which are true in this world are:
{Is_cube(B1), Is_cube(B2), Is_pyramid(P1), Is_pyramid(P2), On_table(B1), On_table(B2), On_table(P1), On(P2,B1).}

The wff

$$\begin{aligned}&\text{Is\_cube}(B1) \wedge \text{Is\_cube}(B2) \wedge\\ &\text{Is\_pyramid}(P1) \wedge \text{Is\_pyramid}(P2) \wedge\\ &\text{On\_table}(B1) \wedge \text{On\_table}(B2) \wedge\\ &\text{On\_table}(P1) \wedge \text{On}(P2,B1)\end{aligned}$$

is satisfied by only the state illustrated above.

Note that

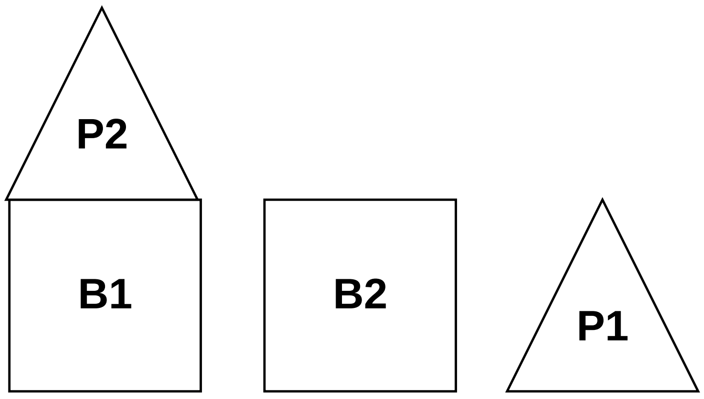
$$\begin{aligned}&\text{On\_table}(B1) \wedge \text{On\_table}(B2) \wedge\\ &\text{On\_table}(P1) \wedge \text{On}(P2,B1)\end{aligned}$$

and even

$$\text{On\_table}(B1) \wedge \text{On\_table}(P1) \wedge \text{On}(P2,B1)$$

Suffice to identify the state uniquely.

It must be emphasized that this is a consequence of the modelling process, and is not true in general. For example, suppose that we drop the propositions of the form On_table(x) from the language. We can still model this blocks world example:



The formula

$$\text{On}(P2,B1) \wedge \neg\text{On}(P1,B2) \wedge \neg\text{On}(B1,B2)$$

describes only the above state. However, no set of propositions describes it.

- In general, to describe a single state, both the propositions which are true and the propositions which are false must be identified.

- In some infinite situations (*e.g.,* any logic which describes the natural numbers **N** = {0, 1, 2, …} with addition), it is impossible to describe each unique possible world, even with propositions and their negations.

Semantic Entailment:

Let W denote the set of the 13 possible worlds in our simple blocks environment. Let φ be any wff. Write

$$\models^{W} \varphi$$

to denote that φ is true in every element of W*; i.e.,* in every possible world. This operation is called *semantic entailment.*

Example: $\models^{W}$ (On_table(B1) $\vee$ On_table(B2))

Now let Ψ be a set of wff's. Write

$$\Psi \models^{W} \varphi$$

to denote that φ is true in every element of W in which every element of Ψ is true. This is also called *semantic entailment*.

Example:

{(On(B1,B2)} $\models^{W}$ (On_table(P1) $\vee$ On_table(P2))

Note that

$\models^{W}$ (On(B1,B2) $\rightarrow$ (On_table(P1) $\vee$ On_table(P2))

also holds. The relationship between the above two statements is in illustration of the *deduction theorem*, which will be discussed in detail later.

Reasoning about the blocks world

Here is what we have so far:

1. A set of possible worlds.
2. A language for describing properties of the possible worlds.

What is missing?

- A means to *reason* about properties of the possible worlds.

Example: Consider the following statement: Either B1 is on the table, or else B1 is on B2. This may be represented by the formula

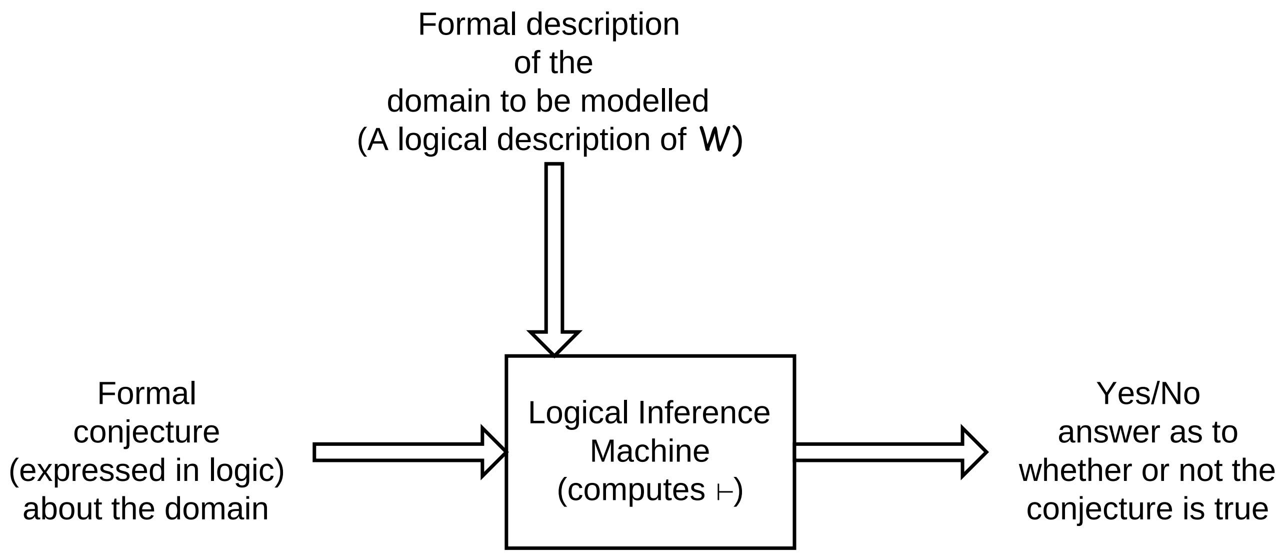
$$\text{On\_table}(B1) \vee \text{On}(B1,B2).$$

To verify the validity of this formula, the only means which we have right now is to check the conditions against each of the thirteen possible worlds.

Most reasonable applications will have a huge number of possible worlds, and this approach will not be feasible.

The alternative is to realize an *inference machine*, as illustrated on the next slide.

A formal inference machine:



To realize this arrangement, a set of formulas Ω^W, which provide a logical description of the set of possible worlds, must be found.

The relation $\vdash$ which the inference machine computes has the property that, for any formal conjecture φ,

$$\models^W \varphi \text{ holds if and only if } W \vdash \varphi \text{ holds.}$$

Key points:

- The relation $\models^W$ is defined by the semantics of the domain to be modelled. It cannot be computed directly, but must be checked against the possible worlds.
- The relation $\vdash$ is computed by the inference engine.

The focus of this course, in a nutshell, is the design of such a logical inference machine.

Classes of Logics

In this course, we will study two classes of logics:

- Propositional logic
- First-order logic

In *propositional logic*, the basic assertions are parameterless assertions. The blocks-world logic which has been presented is basically a propositional logic.

Advantages of propositional logic:

- Simple.
- No decidability problems.

Disadvantages of propositional logic:

- Limited representational power.
- Simple statements may require large and awkward representations.

Example: Consider the statement “There is a stack of three objects.” There are four ways in which this can happen. The formula is:

$(On(B1,B2) \wedge (On(P1,B1) \vee On(P2,B1)) \vee$
$(On(B2,B1) \wedge (On(P1,B2) \vee On(P2,B2)$

Example: Consider the statement “There are at least two objects on the table.” This statement is always true, but to represent it in the propositional form of the logic is awkward, requiring explicit statement of each of the six possibilities.

$(On\_table(B1) \wedge On\_table(B2)) \vee$
$(On\_table(B1) \wedge On\_table(P1)) \vee$
$(On\_table(B1) \wedge On\_table(P2)) \vee$
$(On\_table(B2) \wedge On\_table(P1)) \vee$
$(On\_table(B2) \wedge On\_table(P2)) \vee$
$(On\_table(P1) \wedge On\_table(P2))$

First-order predicate logic:

First-order logic admits:

- Variables which may range over domain elements.
- Quantifiers ("for all" and "there exists").

- This results in much more compact and readable representations:

Example: "There is a stack of three objects."

$$(\exists x)(\exists y)(\exists z)(On(y,x) \wedge On(z,y))$$

(This formula works for any number of objects. Consider how complex the propositional version becomes as the number of objects increases.)

Example: "There are at least two objects on the table." This is a bit trickier. A first attempt is as follows:

$$(\exists x)(\exists y)(On\_table(x) \wedge On\_table(y))$$

However, this is not quite correct, because there is nothing to guarantee that x and y are not the same object. To recapture this example satisfactorily, it is necessary to employ a first order logic with equality.

$$(\exists x)(\exists y)(On\_table(x) \wedge On\_table(y) \wedge (x \neq y))$$

Equality complicates the logic substantially, particularly from a computational point of view.

Advantages of first-order logic:

- Powerful representational capability.
- Many problem cannot be modelled adequately using propositional logic.

Disadvantages of first-order logic:

- Difficult computational problems.
- Inference is undecidable.

Describing the blocks world using logic:

Identifying a set of axioms which characterize the set of possible worlds is often far from trivial, but is nonetheless an essential component of modelling in computer science. Here is an attempt at such a characterization for the simple blocks world.

Everything is either a block or a pyramid:
$(\forall x)(\text{Is\_cube}(x) \vee \text{Is\_pyramid}(x))$

Nothing is both a block and a pyramid:
$(\forall x)(\neg(\text{Is\_cube}(x) \wedge \text{Is\_pyramid}(x)))$

Domain closure; the only objects are those which are identified explicitly:
$(\forall x)(\text{Is\_cube}(x) \leftrightarrow (x=B1 \vee x=B2))$
$(\forall x)(\text{Is\_pyramid}(x) \leftrightarrow (x=P1 \vee x=P2))$

Objects are distinct:
$(B1 \neq B2) \wedge (P1 \neq P2) \wedge (B1 \neq P1) \wedge (B1 \neq P2)$
$\wedge (B2 \neq P1) \wedge (B2 \neq P2)$
(Only the first two statements are necessary, strictly speaking, since the others are deducible.)

No object can rest atop a pyramid.
$(\forall x)(\forall y)(\neg(\text{Is\_pyramid}(x) \wedge \text{On}(y,x)))$

No object can rest atop another object and lie on the table at the same time.
$(\forall x)(\forall y)(\neg(\text{On\_table}(x) \wedge \text{On}(x,y)))$

Every object is either on the table or else atop another object.
$(\forall x)(\exists y)(\text{On\_table}(x) \vee \text{On}(x,y)))$

No object can rest atop itself.
$(\forall x)(\neg \text{On}(x,x)))$

An object can rest atop at most one object.
$(\forall x)(\forall y)(\forall z) ((\text{On}(x,y) \wedge \text{On}(x,z)) \rightarrow y=z))$

At most one object can rest atop another object.
$(\forall x)(\forall y)(\forall z) ((\text{On}(y,x) \wedge \text{On}(z,x)) \rightarrow y=z))$

Remarks:

Is this representation complete? Can “illegal” states sneak in? This is a bit nontrivial even for this very simple setting, and is extremely nontrivial for more complex situations.

If the inference engine is to reason correctly, it must be provided with a complete and correct axiomatization of the set of possible worlds.

These characterizations could also be made in propositional logic, but there would be many more axioms.

Final remarks on logics:

There are many other classes of logics which are of importance in computer science.

- Modal logics
- Temporal logics
- Dynamic logics
- Multi-valued logics
- Fuzzy logics
- Feature logics
- Higher-order logics

To understand these logics, one must first have a thorough understanding of the two basic forms of logic to be covered in this course.