Predicate Logic

Yulei Sui

University of Technology Sydney, Australia

Propositional and Predicate Logic

The following will be the background to understand some terms when using a theorem prover (SAT/SMT solver).

If you have already learned basic discrete math and logic theory, you can skip this. :)

Discrete Math and Basic Logic Theory

Strongly suggest you revisit or pick up discrete math if you haven't. You can learn through the following learning materials or search google for more materials:

- Discrete mathematics wiki
  `https://en.wikipedia.org/wiki/Discrete_mathematics`
- Discrete math videos:
  `https://www.youtube.com/hashtag/discretemathematicsbyneso`
- Propositional logic
  `https://en.wikipedia.org/wiki/Propositional_calculus`
- Predicate logic (or first-order logic)
  `https://en.wikipedia.org/wiki/First-order_logic`
- Discrete mathematics book
  `http://discrete.openmathbooks.org/dmoi3.html`

Satisfiability Solving as Logic Inference Problem

Logic Inference Problem:

- Given:
    - A knowledge base *KB* (a set of constraints (logical formulas) extracted from code statements) and an assertion *Q*, For example,

Satisfiability Solving as Logic Inference Problem

Logic Inference Problem:

- Given:
  - A knowledge base *KB* (a set of constraints (logical formulas) extracted from code statements) and an assertion *Q*, For example,
  - $KB : ((\mathrm{x} > \mathrm{z}) \land (\mathrm{y} == \mathrm{x} + 1)) \lor ((\mathrm{x} \leq \mathrm{z}) \land (\mathrm{y} == 10))$
  - $Q : \mathrm{y} \geq \mathrm{x} + 1$
- $KB \vdash Q$?
  - Does KB semantically entail *Q*?
  - If all constraints in *KB* are true, is the assertion true?
  - Is the specification *Q* satisfiable given constraints from code?
- Each element (**proposition** or **predicate**) in *KB* can be seen as a **premise** and *Q* is the **conclusion.**

Propositional Logic (Statement Logic) [1]

A **proposition** is a statement that is either true or false. Propositional logic studies the ways statements can interact with each other.

- **Propositional variables** (e.g., P and Q) represent propositions or statements in the formal system.
- A **propositional formula** is logical formula with **propositional variables** and **logical connectives** like `and` ($\wedge$) , `or` ($\vee$), `negation` ($\neg$), `implication` ($\Rightarrow$)
- **Inference rules** allow certain formulas to be derived. These derived formulas are called **theorems** (or true propositions). The derivation can be interpreted as proof of the proposition represented by the theorem.

[1] https://en.wikipedia.org/wiki/Propositional_calculus
http://discrete.openmathbooks.org/dmoi2/sec_propositional.html

Propositional Logic (Natural Language Example)

- A natural language inference example where **both premises and conclusion are propositions** in the form of natural language statements.

Premise 1	: If you get 85 marks, then you get a high distinction.
Premise 2	: You get 85 marks.
Conclusion	: You get a high distinction.

Propositional Logic (Natural Language Example)

- A natural language inference example where **both premises and conclusion are propositions** in the form of natural language statements.

Premise 1	: If you get 85 marks, then you get a high distinction.
Premise 2	: You get 85 marks.
Conclusion	: You get a high distinction.

- Propositional variable representation of the above inference rule (P is interpreted as “you get 85 marks“ and Q is “you get a high distinction“)

Premise 1	$P \rightarrow Q$
Premise 2	P
Conclusion	Q

- The inference rule: for any P and Q, whenever $P \rightarrow Q$ and P are true, necessarily Q is true, written in Modus ponens form: $\{P, P \rightarrow Q\} \vdash Q$

Propositional Logic (Code Example)

- A code example where **both premises and conclusion are propositions** in this inference rule.

Premise 1	: if(x $>$ 10 && y $<$ 5) z = 15;
Premise 2	: x $>$ 10;
Premise 3	: y $<$ 5;
Conclusion	: z = 15;

Propositional Logic (Code Example)

- A code example where **both premises and conclusion are propositions** in this inference rule.

Premise 1	: if(x $>$ 10 && y $<$ 5) z = 15;
Premise 2	: x $>$ 10;
Premise 3	: y $<$ 5;
Conclusion	: z = 15;

- Propositional variable representation of the above inference rule (P_1 is interpreted as “x $>$ 10“, P_2 is “y $<$ 5“ and Q is “z = 15“)

Premise 1	: $(P_1 \wedge P_2) \rightarrow Q$
Premise 2	: P_1
Premise 3	: P_2
Conclusion	: Q

- Succinctly written as: $\{P_1, P_2, (P_1 \wedge P_2) \rightarrow Q\} \vdash Q$

Limitations of Propositional Logic (Natural Language Example)

The following valid argument **can** be inferred using propositional logic, i.e., $\{P, P \rightarrow Q\} \vdash Q$

- $P \rightarrow Q$: If you get 85 marks, then you get a high distinction.
- P : You get 85 marks.
- Q : You get a high distinction.

The following valid argument **can not** be inferred using propositional logic, i.e., $\{P', P \rightarrow Q\} \nvdash Q'$ since propositions P' and P are not interpreted as the same.

- $P \rightarrow Q$: If you get 85 marks, then you get a high distinction.
- P' : Jack gets 85 marks.
- Q' : Jack gets a high distinction.

Propositional Logic is **less expressive** and has **weak generalization power**. It does not allow us to conclude the truth of ALL or SOME statements. It is not possible to mention properties of objects in the statement, or relationships between properties.

Limitations of Propositional Logic (Code Example)

The following valid argument **can** be inferred using propositional logic, i.e.,$\{P_1, P_2, (P_1 \land P_2) \rightarrow Q\} \vdash Q$

- P_1: x $>$ 10;
- P_2: y $<$ x;
- $(P_1 \land P_2) \rightarrow Q$: if(x $>$ 10 && y $<$ x) z = 15;
- Q: z = 15;

The following valid argument **can not** be inferred using propositional logic, i.e., $\{P'_1, P'_2, (P_1 \land P_2) \rightarrow Q\} \nvdash Q$

- P'_1: x $=$ 11;
- P'_2: y $=$ 10;
- $(P_1 \land P_2) \rightarrow Q$: if(x $>$ 10 && y $<$ x) z = 15;
- Q: z = 15;

Predicate Logic (First-Order Logic) [2]

First-order logic is propositional logic with predicates and quantification.

- Propositional logic: boolean logic which represents statements without reflecting their structures and relations
- Predicate logic: is more expressive and further analyzes proposition(s) by representing their entities' properties and relations and to group entities, i.e., additionally covers predicates and quantification.

[2] https://en.wikipedia.org/wiki/First-order_logic https://www.youtube.com/watch?v=ARywou8HLQk

Predicate Logic (First-Order Logic) [2]

First-order logic is propositional logic with predicates and quantification.

- Propositional logic: boolean logic which represents statements without reflecting their structures and relations
- Predicate logic: is more expressive and further analyzes proposition(s) by representing their entities' properties and relations and to group entities, i.e., additionally covers predicates and quantification.
- A **predicate** R takes one or more variables/entities as input and outputs a proposition and has a truth value (either true or false).
    - A statement whose truth value is dependent on variables.
    - For example, in $R(x) : x > 5$, "x" is the variable and "> 5" is the predicate. After assigning x with the value 6, $R(x)$ becomes a proposition $6 > 5$.
- A **quantifier** is applied to a set of entities
    - Universal quantifier $\forall$, meaning all, every
    - Existential quantifier $\exists$, meaning some, there exists

[2] https://en.wikipedia.org/wiki/First-order_logic https://www.youtube.com/watch?v=ARywou8HLQk

Predicate Logic (Natural Language Example)

Consider the two statements

- “Jack got a high distinction”
- “Peter got a high distinction”

In propositional logic, these statements are viewed as being unrelated and the sub-statements/words/entities are not further analyzed.

- **Predicate logic** allows us to define a **predicate** R representing “got a high distinction” which occurs in both sentences.
- $R(x)$ is the **predicate logic statement (formula)** which accepts a name x and output as “x got a high distinction”.

Predicate Logic (Code Example)

- P_1: x $>$ 10;
- P_2: y $<$ x;
- $(P_1 \wedge P_2) \rightarrow Q$: if(x $>$ 10 && y $<$ x) z = 15;
- Q: z = 15;

In propositional logic, each code statement (including its variables) is viewed as one proposition. Its variables and their relations are not further analyzed.

- **Predicate logic** allows us to define the following three predicates
  - $R_1(x)$ representing x $>$ 10 for the property of a single variable.
  - $R_2(y, x)$ representing y $<$ x for the relation between two variables.
  - $R_3(z)$ representing z = 3 for the property of a single variable.

Predicate Logic (Code Example)

- P_1: x $>$ 10;
- P_2: y $<$ x;
- $(P_1 \wedge P_2) \rightarrow Q$: if(x $>$ 10 && y $<$ x) z = 15;
- Q: z = 15;

In propositional logic, each code statement (including its variables) is viewed as one proposition. Its variables and their relations are not further analyzed.

- **Predicate logic** allows us to define the following three predicates
  - $R_1(x)$ representing x $>$ 10 for the property of a single variable.
  - $R_2(y, x)$ representing y $<$ x for the relation between two variables.
  - $R_3(z)$ representing z = 3 for the property of a single variable.

Given **a set of propositions/predicates** as the knowledge base *KB*, let us take a look at how we do **the inference** $KB \vdash Q$ using propositional logic and predicate logic. Does *KB* semantically entail Q?

Predicate Logic (Code Example)

- P_1: x $>$ 10;
- P_2: y $<$ x;
- $(P_1 \wedge P_2) \rightarrow Q$: if(x $>$ 10 && y $<$ x) z = 15;
- Q: z = 15;

In propositional logic, each code statement (including its variables) is viewed as one proposition and its variables are not further analyzed.

- **Predicate logic** allows us to define the following three predicates
  - $R_1(x)$ representing x $>$ 10 for relation/property of a single variable.
  - $R_2(y, x)$ representing y $<$ x for relation between two variables.
  - $R_3(z)$ representing z = 3 for relation/property of a single variable.

Predicate Logic (Code Example)

- P_1: x $>$ 10;
- P_2: y $<$ x;
- $(P_1 \wedge P_2) \rightarrow Q$: if(x $>$ 10 && y $<$ x) z = 15;
- Q: z = 15;

In propositional logic, each code statement (including its variables) is viewed as one proposition and its variables are not further analyzed.

- **Predicate logic** allows us to define the following three predicates
  - $R_1(x)$ representing x $>$ 10 for relation/property of a single variable.
  - $R_2(y, x)$ representing y $<$ x for relation between two variables.
  - $R_3(z)$ representing z = 3 for relation/property of a single variable.

Propositional logical for the inference

- $\{P_1, P_2, (P_1 \wedge P_2) \rightarrow Q\} \vdash Q$

Predicate logical for the inference

- $\{R_1(x), R_2(y, x), (R_1(x) \wedge R_2(y, x)) \rightarrow R_3(z)\} \vdash Q$

Predicate Logic (Code Example)

- P_1: x $>$ 10;
- P_2: y $<$ x;
- $(P_1 \wedge P_2) \rightarrow Q$: if(x $>$ 10 && y $<$ x) z = 15;
- Q: z = 15;

In propositional logic, each code statement (including its variables) is viewed as one proposition and its variables are not further analyzed.

- **Predicate logic** allows us to define the following three predicates
  - $R_1(x)$ representing x $>$ 10 for relation/property of a single variable.
  - $R_2(y, x)$ representing y $<$ x for relation between two variables.
  - $R_3(z)$ representing z = 3 for relation/property of a single variable.

Propositional logical for the inference

- $\{P_1, P_2, (P_1 \wedge P_2) \rightarrow Q\} \vdash Q$

Predicate logical for the inference

- $\{R_1(x), R_2(y, x), (R_1(x) \wedge R_2(y, x)) \rightarrow R_3(z)\} \vdash Q$
- $\{x > 10, y < x, (x > 10 \wedge y < x) \rightarrow z = 3\} \vdash z = 3$

Verification as Solving Constrained Horn Clauses

A constrained horn clause has the following form:

- $\forall V \; (\varphi \wedge R_1(X_1) \wedge \dots \wedge R_k(X_k)) \rightarrow Q(X)$
    - τ is a **background theory** (e.g., Linear Arithmetic, Arrays, BitVectors, or combinations of the above)
    - V are variables, and X_i is a set of **terms** over V
    - φ is a constraint in the background theory τ
    - $R_1, \dots R_k$ and Q are multi-ary **predicates**.
    - $R_i(X_i)$ is an application of predicate R_i to **first-order terms**.

Verification as solving constrained horn clauses.