Week 2-3: Propositional Logic

Wang Xiyu

Propositional logic is about, well, logics about propositions. A prposition is basically a statement that can be evaluated to true or false. For example "1+1=2" and "1+1=3" are both propositions.

1 Syntax

Definition 1.1. *Syntax decides what are allowed to be written*

1.1 Well-formed Formula (WFF)

A formula ψ is obtained by composing $p \in P$ with $c \in C$, where P is the set of propositions and C is the set of connectors. here we use any set of complete logic, like $C = \{\neg, \vee\}$. This is not a percise definition as counter examples that do not form a valid formula can be formulated based on this, for example $\neg \vee p$ means nothing. To formalize the definition we may attempt to define a language with the following grammar:

$$\Sigma = P \cup C \cup \{(,)\}$$

$$FORM : \{\psi = p \in P | \neg\psi | \psi \vee \psi | (\psi)\}$$

However this is a recursive definition as the definition of the a formula ψ involves itself. We need to find a declarative way to define this recursive/inductive construction.

Definition 1.2. *FORM is* ***the smallest*** *set of strings over Σ where*

1. $P \subseteq FORM$
2. $\forall \psi \in FORM, \neg\psi \in FORM$
3. $\forall \psi_1, \psi_2 \in FORM, \psi_1 \vee \psi_2 \in FORM$

We can see as a construction rule, we are admitting strings into the set of formula, but we need to somewhat impose restriction on what is not acceptable.

1.2 Closed Set

Let's remove the attributive **the smallest** first and define another set that contains all the set that satisfy these rules listed above:

Definition 1.3. *Closed is a set of strings over Σ where*

1. $P \subseteq Closed$
2. $\forall \psi \in Closed, \neg\psi \in Closed$
3. $\forall \psi_1, \psi_2 \in Closed, \psi_1 \vee \psi_2 \in Closed$

1.2.1 All Closed Set

And we define a set ACS (All-closed sets). We can define FORM via the following constriction:

$$FORM = \bigcap ACS$$

Suppose $BCS = \emptyset$ and from definition

$$U_0 = \bigcap BCS = \{w \in \Sigma^* | \forall S \in BCS, w \in S\}$$

However, $BCS = \emptyset$, The statement $\forall S \in BCS, w \in S$ is vacuously true $\forall w \in \Sigma^*$, which means that

$$U_0 = \Sigma^* = \bigcap BCS$$

A contradiction. Therefore we have proved that ACS cannot be empty, thus $\forall P, \exists S$ such that S is a closed set.

1.3 FORM is the minimum of the set of Closed

First let's prove that FORM is closed. It seems to be trivial that the intersection of closed sets is closed but formal proof is still needed. We denote a set from the intersection construction as T_0.

Lemma 1.4. *T_0 is closed*

We are proving that the set T_0 we get from the intersection construction still satisfy the rules that define what is closed.

Proof. From rule 1,

$$\begin{aligned}
&\forall S \in ACS, P \subseteq S \\
&\quad \implies \forall p \in P, \forall S \in ACS, p \in S \\
&\quad \implies \forall p \in P, p \in \bigcap ACS \\
&\quad \implies P \in T_0
\end{aligned}$$

From rule 2,

$$\begin{aligned}
&\forall S \in ACS, T_0 \subseteq S \\
&\text{let } w \in T_0, \forall S \in ACS, w \in S, \neg w \in S \\
&\quad \implies \forall w \in T_0, \neg w \in \bigcap ACS \\
&\quad \implies \forall w \in T_0, \neg w \in T_0
\end{aligned}$$

From rule 3,

$$\begin{aligned}
&\forall S \in ACS, T_0 \subseteq S \\
&\text{let } w_1, w_2 \in T_0, \forall S \in ACS, w_1 \vee w_2 \in S \\
&\quad \implies \forall w_1, w_2 \in T_0, w_1 \vee w_2 \in \bigcap ACS \\
&\quad \implies \forall w_1, w_2 \in T_0, w_1 \vee w_2 \in T_0
\end{aligned}$$

□

Lemma 1.5. *T_0 is **a** smallest Closed set*

Proof. The proof of minimality is trivial. $T_0 = \bigcap ACS \implies \forall S \in ACS, T_0 \subseteq S$ □

Lemma 1.6. *T_0 is **the** smallest Closed set (unique). In other word,*

$$\forall T_0' \in \{T | \forall S \in ACS, T \subseteq S\}, T_0' = T_0$$

Proof. Suppose otherwise, $\exists T_0' \in ACS \neq T_0$.

$$\begin{aligned} T_0, T_0' \in ACS &\implies \forall S \in ACS, T_0 \subseteq S, T_0' \subseteq S \\ &\implies T_0 \subseteq T_0' \wedge T_0' \subseteq T_0 \\ &\implies T_0 = T_0' \end{aligned}$$

A contradiction. □

Now we can use the generative rule to define formula, which is called **Backus-Naur form**. Note that a context-free grammar is a type of formal language. Backus Naur form is a specification language for this type of grammar. It is used to describe language syntax.

1.4 Subformula

Definition 1.7. *ϕ_1 is a subformula of ϕ_2, denoted by*

$$\phi_1 \preccurlyeq \phi_2$$

as a relation is

- *reflexive:* $\forall \phi \in FORM, \phi \preccurlyeq \phi$
- *transitive:* $\forall \phi_1, \phi_2, \phi_3 \in FORM, (\phi_1 \preccurlyeq \phi_2) \wedge (\phi_2 \preccurlyeq \phi_3) \implies \phi_1 \preccurlyeq \phi_3$

Inductive definition:

$$\begin{aligned} \forall \phi \in FORM, \phi &\preccurlyeq (\neg \phi) \\ \forall \phi_1, \phi_2 \in FORM, \phi_1 &\preccurlyeq (\phi_1 \vee \phi_2) \\ \phi_2 &\preccurlyeq (\phi_1 \vee \phi_2) \end{aligned}$$

We might intuitively think that subformula can be defined using the notion of substring, since we have been considering formula as concatenation of proposition and connector symbols. However based on out inductive definition, we can come up with the following counter example to a definition using substring: A formula ϕ_1 is a subformula of ϕ_2 if and only if ϕ_1 is a well-formed formula, and is a substring to ϕ_2

$$p \vee q \not\preccurlyeq \neg p \vee q$$

What if we tighten the syntax by strictly enforce the use of parenthese whenver we can

2 Sematics

Definition 2.1. *Sematics describe meanings.*

2.1 Truth Assignment/Proposition evaluation

A truth assignment or a valuation of propositions is a function that maps from the set of propositions to true and false.

$$v : P \to \{T, F\}$$

$$v \in [P \to \{T, F\}]$$

Example 2.2. *Consider $v = \{p_1 \mapsto T, p_2 \mapsto F\}$, we can derive*

$$\begin{aligned}
&v(p_1) = T \\
&v(p_2) = F \\
&v(\neg p_2) = T \\
&v(p_1 \vee p_2) = T \ \textit{etc.}
\end{aligned}$$

2.2 Entail/Model

We denote entailment of an evaluation function to formula as

$$v \models \psi$$

We also indeuctively define the following rule

$$\begin{aligned}
\forall p \in P, v \models p &\iff v(p) \\
v \models \psi &\iff v \not\models \neg\psi \\
v \models \psi_1 \vee \psi_2 &\iff (v \models \psi_1) \vee (v \models \psi_2)
\end{aligned}$$

2.3 Logical Equivalence

Definition 2.3. *Syntax definition of logical equivalence is as follows:*

$$\phi_1 \equiv \phi_2 \iff \forall v : P \mapsto \{T, F\}(v \models \phi_1), v \models \phi_2$$

Example 2.4.

$$p \vee (\neg p) \equiv q \vee (\neg q)$$

$$\begin{aligned}
&\forall v \models (p \vee (\neg p)), \\
&v \models p \ \textit{or} \ v \models (\neg p) \\
&v \models p \ \textit{or} \ v \not\models p \\
&\forall v : P \mapsto \{T, F\},
\end{aligned}$$

TODO

2.4 Satisfiability of Formula

Definition 2.5. *Let $\phi \in FORM, \phi$ is satisfiable*

$$\iff \exists v : P \mapsto \{T, F\}, v \models \phi$$

Definition 2.6. *Tautology: ϕ is a tautology $\iff \forall v, v \models \phi$*

Example 2.7.

$$\phi = \neg(p \vee (\neg p))$$

From example 2.4, we have

$$\begin{aligned}
&\forall v, v \models (p \vee (\neg p)) \\
&\implies \forall v, v \not\models \neg(p \vee (\neg p))
\end{aligned}$$

Thus ϕ is NOT satisfiable.

2.5 Satisfiability of a set of formula

Set Satisfication

Definition 2.8. *Let* $\Gamma \subseteq FORM$,

$$v \models \Gamma \iff \forall \phi \in \Gamma, v \models \phi$$

Set Satisfiability

Definition 2.9. *Let* $\Gamma \subseteq FORM$, Γ *is satisfiable*

$$\iff \exists v, \forall \phi \in \Gamma, v \models \phi$$

Example 2.10. $\Gamma = \emptyset$, Γ *is satisfiable.*

Proof. Consider $v : P \mapsto \{T, F\}, v(p) = T \forall p \in P$, $\forall \phi \in \Gamma, v \models \phi$ is vacuously true. □

Example 2.11. $\Gamma = FORM$, Γ *is NOT satisfiable.*

Proof. Suppose otherwise,

$$\begin{aligned} \exists v, \forall \phi \in \Gamma, v \models \phi \\ \forall \phi \in \Gamma, (\neg \phi) \in \Gamma \\ v \models (\neg \phi) \end{aligned}$$

A contradiction. □

2.5.1 Relevance

Definition 2.12. *Define set* $OCCUR(\phi) \subseteq P$, *which contains propositions appear in* ϕ. *Inductively defined as*

$$\begin{aligned} &OCCUR(\phi) = \{p\} \iff \phi \equiv p, p \in P \\ &OCCUR(\neg\phi) = OCCUR(\phi) \\ &OCCUR(\phi_1 \vee \phi_2) = OCCUR(\phi_1) \cup OCCUR(\phi_2) \end{aligned}$$

Lemma 2.13. *Let* $\phi \in FORM$, *let* v_1, v_2 *be valuations over* P.

$$\forall p \in OCCUR(\phi), v_1(p) = v_2(p) \implies (v_1 \models \phi \implies v_2 \models \phi)$$

v_1, v_2 *can be different valuations over* $p' \notin OCCUR(\phi)$

2.6 Modelling Computations as Propositional Logic

Computation problems can be considered as membership check of an encoded problem instance to a language.

$$L = \{p | \langle M, p \rangle\}$$

Here instead of arbituary encoding, we consider problem encoding with propositional logics only, and solve with SAT solvers by deciding whether the set of formula we form from any instance of the problem is satisfiable. That is

$$Encode : p \mapsto FORM, \forall p, \exists v \models Encode(p) \iff p \in L$$

The rules of such modelling include:

- Propositional logics only
- The final expression must one well formed fomula
- for problem of size n, the size of the formula ϕ must be in $O(poly(n))$

Note that there can be different definition on the size of the

- problem: in unary, binary etc;
- Formula: Whether by the number of nodes in the syntax tree or length of the string Anything else?

Example 2.14. *Graph coloring problem: given a undirected graph $G = (V, E)$, whether there exists a k coloring of all vertices such that no neighbouring vertices have the same color.*
Consider the following construction of a proposition set P:

$$P = \{p_{v,i} | v \in V, i \in [1, k]\}$$

The valuation of proposition $p_{v,i}$ conceptually represents vertex v is assigned color i.
We can formulate constriants as well formed formula, and use conjunction to connect the formula to model satisfaction of multiple constraints.

Corollary 2.15. $\forall \phi_1, \phi_2 \in FORM, (\phi_1 \wedge \pi_2) \in FORM$

First let us list down the constraints of this problems, including implicit ones in plain words

1. *Neighbouring vertices have different colors.*
2. *All vertices as assigned with exactly one color.*

1. ϕ_{nbr}

First lets formula the constraint between any 2 vertices,

$$\forall (\alpha, \beta) \in E, c(v_\alpha) \neq c(v_\beta)$$

And from our proposition set, $c(v_\gamma) = i \iff \exists v, v \models p_{\gamma,i}$. As such we have

$$\phi_{\alpha,\beta,i} = \neg(p_{\alpha,i} \wedge p_{\beta,i})$$

This models vertex α and vertex β are not colored with color i at the same time. For every color this needs to hold, thus we have

$$\phi_{nbr(\alpha,\beta)} = \bigwedge_{i \in [1,k]} \phi_{\alpha,\beta,i}$$

For every pair of neighbouring vertices this also needs to hold. Therefore we have

$$\phi_{nbr} = \bigwedge_{(\alpha,\beta) \in E} \phi_{nbr(\alpha,\beta)}$$

2. $\phi_{exactly-1}$

To model exactly 1 coloring on each vertex, we can formulate $\phi_{\geq 1}$ *and* $\phi_{\leq 1}$ *first and use conjunction to connect them. Given our propositions, to make sure vertex* v *is at least assign a color* ($\phi_{v,\geq 1}$),

$$\phi_{v,\geq 1} = \bigvee_{i \in [1,k]} p_{v,i}$$

This needs to hold for every vertex, thus we have

$$\phi_{\geq 1} = \bigwedge_{v \in V} \phi_{v,\geq 1}$$

Then we need to model at most 1, (or less than 2) coloring on each vertex. Vertex v *is not assignment with both color* i *and* j *can be formulated as*

$$\phi_{v,i,j,\leq 1} = \neg(p_{v,i} \wedge p_{v,j})$$

This needs to hold for every distinct pair of colors, thus

$$\phi_{v,\leq 1} = \bigwedge_{i \neq j \in [1,k]} \phi_{v,i,j,\leq 1}$$

For all vertices

$$\phi_{\leq} = \bigwedge_{v \in V} \phi_{v,\leq}$$

Finally we have

$$\phi_{exactly-1} = \phi_{\leq 1} \wedge \phi_{\geq 1}$$

And the overall formula for an instance of $G = (V, E)$ *becomes*

$$\phi = \phi_{nbr} \wedge \phi_{exactly-1}$$

Similar to reduction, we need also now to show that

$$\exists v : P \mapsto \{T, F\}, v \models \phi \iff \langle G = (V, E), k \rangle \in L_{color}$$

TODO

Tutorial

2.7 Generic closure

* Define $\mathcal{P}(U)$ as a collection of all ordering of all elemenst in the power set $\mathscr{P}(U)$

Definition 2.16. *Given a universe set U and an operator set $\mathcal{O}$, where $\forall f : U \times r \text{ times } \times U \mapsto \mathcal{P}(U) \in \mathcal{O}$. f has arity $r = arity(f)$. We define a set $S \subseteq U$ to be (I, O)-closed if the following hold:*

1. $I \subseteq S$
2. $\forall f \in O, e_1, e_2, \ldots, e_r \in I, f(e_1, e_2, \ldots, e_r) \in S$

Lemma 2.17. *There exists a smallest (I, O)-closed set.*

Q1

Claim: The smallest (I, O)-closed set can be constructed by taking the intersection of all (I,O)-closed set, ie.

$$AIOCS = \{S \subseteq U | S \text{ S is (I,O)-closed}\}$$

$$T_0 = \bigcap AIOCS$$

Existence

First we prove the existence of such a set.

Proof. Suppose otherwise,

$$T_0 = \bigcap AIOCS = \emptyset$$
$$T_0 = \{x \in \mathcal{P}(U) | \forall S \in AIOCS, x \in S\}$$
$$T_0 = \emptyset \implies x \in S \text{ is vacuously true } \forall x \in \mathcal{P}(U) \implies T_0 = \mathcal{P}(U)$$

contradiction. □

Closed

Now we prove that T_0 is (I,O)-closed.

Proof. Property 1:

$$\forall S \in AIOCS, I \subseteq S$$
$$\implies \forall i \in I, \forall S \in AIOCS, i \in S$$
$$\implies \forall i \in I, i \in \bigcap AIOCS$$
$$\implies \forall i \in I, i \in T_0$$

Property 2:

$$\forall S \in AIOCS, T_0 \subseteq S$$
$$\text{Let } e_1, e_2, \ldots, e_r \in T_0, \forall S \in AIOCS, e_1, e_2, \ldots, e_r \in S$$
$$\implies \exists f \in \mathcal{O}, f(e_1, e_2, \ldots, e_r) \in S$$
$$\implies \forall e_1, e_2, \ldots, e_r \in T_0, \exists f \in \mathcal{O}, f(e_1, e_2, \ldots, e_r) \in \bigcap AIOCS$$
$$\implies f(e_1, e_2, \ldots, e_r) \in T_0$$

□

Uniqueness

Claim:

$$\forall T_0' \in \{T \in \mathcal{P}(U) | \forall S \in AIOCS, T \subseteq S\}, T_0' = T_0$$

Proof. Suppose otherwise, $\exists T_0' \in AIOCS, T_0' \neq T_0$.

$$\begin{aligned} T_0, T_0' \in AIOCS &\implies \forall S \in AIOCS, T_0 \subseteq S, T_0' \subseteq S \\ &\implies T_0 \subseteq T_0', T_0' \subseteq T_0 \\ &\implies T_0 = T_0' \end{aligned}$$

□

Q2

2.1

$$w_1 = \epsilon \notin FORM$$

Claim: $\forall w \in FORM, |w| > 0$

Proof. Base case:

$$\forall p \in P, |p| = 1$$

Inductive Hypothesis:

□

Q3

1. $\mathbb{N} : I = \{0\}, O = \{f(x) = x + 1\}$
2. $Odd : I = \{1\}, O = \{f(x) = x + 2\}$
3. $\mathbb{Q} : I = \{0\}, O = \{f(x) = -x, f(x) = x + 1, f(x, y) = \frac{x}{y}\}$
4. $\{\frac{x}{3} | x \in \mathbb{N}\} \setminus \mathbb{N} : I = \{\frac{1}{3}\}, O = \{f(x) = 3x + 1 | 3?\frac{1}{3} : x + \frac{1}{3}\}$

Q4

4.1

$I = \{t\}, O = \{f(v) = succ(v)\}$

4.2

$I = \{s\} \cup \{v \in V | v \in succ^r(s)\}, O = \{f(v) = succ^d(v)\}$

Q5

5.1

$I = \{(v, v) | v \in V\}, O = \{f(u, u') = \{(u', v) | v \in V, succ(u') = v\}\}$

Q6

6.1

Base case: $FORM_0 \subseteq FORM_1$ as

$FORM_1 = FORM_0 \cup \{(\neg w) | w \in FORM_0\} \cup \{(w_1 \vee w_2) | w_1, w_2 \in FORM_0\}$

Wait this is just trivial

6.2

Claim:

$$\exists i \geq 0, FORM_i = FORM_{i+1}$$

It is obvious that $\forall i \geq 0, FORM_i$ is finite. Suppose

6.3

6.4

Q7

Claim: $INDFORM$ is (P, C)-closed

Q8

Q9