CMPT 409: Theoretical Computer Science **Summer 2017**

Lecture 1: Introduction (8 - 19 May)

Lecturer: Ternovska, Eugenia *Scribe: Lily Li*

1.1 Propositional Calculus

Again to reiterate: **syntax** has to do with the structure of strings of symbols (e.g. formulas and formal proofs) and the rules for their manipulation. **Semantics** has to do with meaning (e.g. validity).

1.1.1 Syntax

Formulas come in different flavors. First we will discuss **propositional formulas** (i.e. atoms, negations, ands, and ors), later we will talk about **first-order formulas** (i.e. propositional formulas with universal and existential qualifiers).

A **subformula** of a formula A is any substring of A which is also a formula (being a formula is important). We use the following notation:

$\supset$ means *implies* and $A \supset B$ is equivalent to $A \vee \neg B$.

$\leftrightarrow$ means *is equivalent to* and $A \leftrightarrow B$ also stands for $(A \supset B) \wedge (B \supset A)$.

Theorem 1.1 ***Unique Readability****: a.k.a the grammar for generating formulas is unambiguous. Suppose A, B, A', B' are formulas, c, c' are binary connectives, and $(AcB) =_{syn} (A'c'B')$. Then $A =_{syn} A'$, $c =_{syn} c'$, and $B =_{syn} B'$. Where $=_{syn}$ means equality as symbols (not only meaning).*

We will introduce a definition and prove a lemma in order to prove the theorem. The **weight** of a formula A is the sum of the weights of the symbols of A where the symbols are assigned weights:

$$\begin{aligned} &0 \text{ to } \neg \\ &1 \text{ to each binary connective } \wedge, \vee \\ &1 \text{ to } (\\ -&1 \text{ to }) \\ -&1 \text{ to each atom } P \end{aligned}$$

Lemma 1.2 *The weight of any formula is -1, but the weight of any proper initial segment is ≥ 0 (where a* ***proper*** *initial segment is the initial part which is not a formula).*

Proof: **Structural induction** on the length of A. The base case where A is a single atom satisfies the claim. In the inductive step we consider the three ways of forming larger formulas: $\neg, \wedge, \vee$ (only show $\wedge$, others are the same). If $A =_{syn} (P \wedge Q)$, then the weight of A, $w(A) = -1$ since $w(\wedge) = 1$, $w(() = 1$, $w()) = -1$, and $w(P) = w(Q) = -1$ by the induction hypothesis. ■

We prove the **Unique Readability Theorem** using the above lemma. If $A =_{syn} A'$, then we are done. Suppose not. Then either A is a proper initial segment of A' or A' is a proper initial segment of A. WLOG assume the former. Then A has more left than right brackets. But (AcB) is well formed so A has the same number of left and right brackets. This is a contradiction.

The theorem basically says that if you put in all the brackets then the formula has only one interpretation, but in practice we omit most of the brackets and use left associativity.

1.1.2 Semantics

A **truth assignment** is a map $\tau : \{\text{atoms}\} \to \{T, F\}$. It can be extended to map formulas to $\{T, F\}$ by defining $(\neg A)^\tau = T$ if and only if $A^\tau = F$ and similarly for $\wedge$ and $\vee$. A truth assignment τ satisfies A iff $A^\tau = T$; τ satisfies a set Φ of formulas iff τ satisfies A for all $A \in \Phi$.

Φ is satisfiable if there exists some truth assignment τ such that τ satisfies Φ. Otherwise Φ is unsatisfiable. Similarly for A.

Logical Consequence: A is a — of Φ, written $\Phi \vDash A$, iff for every truth assignment τ, if τ satisfies Φ, then τ satisfies A.

We write $\vDash A$ to mean $\emptyset \vDash A$, $B \vDash A$ to mean $\{B\} \vDash A$, and $B, C \vDash A$ to mean $\{B, C\} \vDash A$.

Proposition 1.3 ***Transitivity of Logical Consequences:*** *if $\Phi \vDash A$ and $\Phi \cup \{A\} \vDash B$, then $\Phi \vDash B$.*

Proof: Consider any truth assignment τ. If τ satisfies Φ then τ satisfies A since $\Phi \vDash A$. Thus τ satisfies $\Phi \cup \{A\}$ thus τ satisfies B. ■

A formula A is **valid** iff $\vDash A$ (i.e. $A^\tau = T$ for all τ). A valid propositional formula is a **tautology**. Formulas A, B are equivalent, written $A \longleftrightarrow B$ iff $A \vDash B$ and $B \vDash A$ (this is "equivalent" as you would typically understand the word that is semantic equivalence. This is different from $=_{syn}$ (explain why). Typically it is the convention that P, Q, R stand for *distinct* atoms and $A, B, C, ...$ could stand for identical formulas.

Proposition 1.4 $\Phi \vDash A \iff \Phi \cup \{\neg A\}$ *is unsatisfiable. Also A is a tautology iff $\neg A$ is unsatisfiable.*

Proof: For any truth assignment τ which satisfies Φ, τ does satisfies A since $\Phi \vDash A$. Thus τ cannot satisfy $\Phi \cup \{\neg A\}$. ■

An example of tautologies for formulas A, B:

$$(A \wedge B) \vDash (A \vee B)$$

Theorem 1.5 ***Duality Theorem*** *Let A' be the result of interchanging $\wedge$ and $\vee$ in the formula A, and replacing P by $\neg P$ for each atom P. Then $A' \longleftrightarrow \neg A$.*

Proof: Proof by structural induction on A. If A is just one atom then the claim clearly holds. Consider each method of making a larger formula. If the outer most operation of A is $\neg$ then the claim holds. If the outer most operation is $\wedge$ then the claim holds by DeMorgan's Law (same with $\vee$). ■

Theorem 1.6 ***Craig Interpolation Lemma*** *Given propositional formulas A and B, let S be the set of atoms which occur in both A and B, and assume that S is nonempty. If $A \supset B$ is valid, there is a formula C (an* interpolant*) containing only atoms from S such that both $A \supset C$ and $C \supset B$ are valid.*

Proof:

■

1.1.3 DNF and CNF

A **literal** l is an atom P or its negation. A **clause** C for a CNF is a disjunction ($\vee$) of literals such that no literal occurs twice. A formula is in **conjunctive normal form (CNF)** if it is a conjunction ($\wedge$) of one or more clauses. We consider the empty conjunction $\wedge\emptyset$ a CNF formula, and $\wedge\emptyset$ *is* valid. **Disjunctive normal form** is defined similarly and is the dual notion to CNF. The empty disjunction $\vee\emptyset$ is in DNF and is unsatisfiable.

Theorem 1.7 *Every formula is equivalent to a formula in CNF, and to a formula in DNF.*

Proof: To put a formula A into DNF, simply take every truth assignment satisfying A, form a clause out of it, and $\vee$ all such clauses together. To put A into CNF is quite similar, though you might also consider applying logical equivalences to your DNF. ■

Proposition 1.8 *Every CNF formula equivalent to*

$$(P_1 \wedge Q_1) \vee (P_2 \wedge Q_2) \vee \cdots \vee (P_n \wedge Q_n)$$

must have at least 2^n *clauses.*

Proof: Observe that every clause has n literals, one of each index. Suppose that some CNF formulation A has few than 2^n clauses. Then some clause $c = R_1 \vee \cdots \vee R_n$ where $R_i \in \{P_i, Q_i\}$ is not A. Set $R_1 = \cdots = R_n = F$, and set all the remaining literal to be true. Since c is not in A, each clause in A must differ from c in one literal and must thus be true. Thus A is true. However the formula $(P_1 \wedge Q_1) \vee (P_2 \wedge Q_2) \vee \cdots \vee (P_n \wedge Q_n)$ is false since every R_i is false. Thus A can not be equal to the formula. ■

1.2 Formal Proofs

These are ways of showing that a propositional formula is a tautology. Unlike validity, a semantic notion, formal proofs have to do with syntax.

1.2.1 Resolution

This is one well studied proof system for establishing the unsatisfiability of a set of CNF formulas.

Theorem 1.9 *There is a polynomial time procedure which transforms a given finite set* Φ *of propositional formulas to a finite set* $S = S_\Phi$ *of clauses, such that* Φ *is satisfiable iff* S *is satisfiable.*

Proof: The straight forward trick of putting every formula in CNF then smashing all the CNF's together into one large CNF does not work since there could be an exponential blowup in the number of clauses. Instead what you are suppose to do is to use the standard trick for reducing General Propositional Satisfiability to SAT. Basically you take a formula A and start with a "small" subformula B and create a new atom P_B (if B

is a literal l then let $P_B = l$). *Think of P_B as the name for B even though it is a separate axiom.* Add to the set S all the clauses necessary to link P_B to B. That is the clauses (disjunctions) equivalent to $P_B \longleftrightarrow B$. Repeat, using P_B in place of B. Once you get to the top add a new atom P_A for A, then add the clause P_A to S. See notes for more detail.

A is not equivalent to the conjunction of the clauses in S, but it is satisfiable iff S is satisfiable. To see this non-equivalent consider the following example: take the single formula $A = (Q \wedge R) \vee \neg Q)$ as given in the Cook notes page 7. If we let $B = (Q \wedge R)$ with the added atoms P_B and P_A we get that

$$S = \{\bar{P_B} \vee Q, \bar{P_B} \vee R, P_B \vee \bar{Q} \vee \bar{R}, P_B \vee \bar{P_A} \vee \bar{Q}, P_A \vee \bar{P_B}, P_A \vee Q, P_A\}$$

(the first three mean $P_B \longleftrightarrow (Q \wedge R)$, the next three mean $P_A \longleftrightarrow (P_B \vee \neg Q)$). Thus if we assign $Q = R = T$ but $P_B = F$ the conjunction of S is false while A is true. ■

Now that we have reduced the problem of determining satisfiability of a set of formulas to determining the satisfiability of a set of clauses, we can apply the **resolution rule**: consider clauses $C_1 = (A \vee l)$ and $C_2 = (B \vee \bar{l})$ where A and B are clauses not containing l or $\bar{l}$. Then the **resolvent** of C_1 and C_2 is the clause $C_3 = (A \vee B)$. Remark $(P \vee Q)$ and $(\bar{P} \vee \bar{Q})$ does NOT have a resolvent since there are two clashes.

Resolvent Soundness Principle: If C_3 is the resolvent of C_1 and C_2 then

$$C_1, C_2 \models C_3$$

This holds even if C_3 is the empty clause $\vee\emptyset$ (this proves that C_1 and C_2 are unsatisfiable).

A **resolution refutation** of a set S of clauses is a sequence $C_1, ..., C_q$ of clauses such that the final clause C_q is the empty clause. Each C_i is either in S (these are the axioms) or the resolvent of earlier clauses in the sequence.

Theorem 1.10 ***RES Soundness Theorem:*** *If a set S of clauses has a resolution refutation, then S is unsatisfiable.*

Proof: Let $C_1, ..., C_p$ be the resolution refutation of S. By the Resolvent Soundness Principle and the Transitivity of Logical consequence, the empty clause $\vee\emptyset$ is a logical consequence of S. Since $\vee\emptyset$ is unsatisfiable S must be unsatisfiable as well. ■

Theorem 1.11 ***RES Completeness Theorem:*** *Every unsatisfiable set of clauses has a resolution refutation.*

Proof: Quite a nasty little bugger, this proof is. We will only prove it for finite set S, though with the Propositional Compactness theorem (shown later), the proof can be extended for infinite sets S.

So we will iteratively construct of resolution refutation (if it exists), otherwise we will find a satisfying assignment for S. During the procedure, we maintain a sequence $S' \supset S$ of clauses which forms a partial resolution refutation of S. We also maintain a stack of literals $l_1, l_2, ..., l_k$ and a partial truth assignment τ to the atoms of S. τ makes each literal on the stack true and does not falsify any clause in S'.

1. If S consists of only the empty clause $\vee\emptyset$ then S is unsatisfiable.
2. Suppose that S is non-empty. The iterative procedure is as follows: check if τ satisfies S. If so, output the t.a. τ. Otherwise take a unsatisfied clause $C \in S$ and a unsatisfied literal $l \in C$. Add l to the stack and let τ' be the extension of τ which assigns true to l. If this assignment does not falsify any clauses in S', repeat.

3. Suppose τ' falsifies C' in S'. Replace l with $\bar{l}$ on the stack. Let τ" be the extension of τ which assigns true to $\bar{l}$. If τ" does not falsify any clause in S', then this assignment works. Go back to step 2.
4. Otherwise there exists a clause C" $\in S'$ which is falsified by $\bar{l}$. Resolve C' and C" into clause R which does not contain the l or $\bar{l}$ literals. If R is the empty clause, then $S' \cup \{R\}$ is a resolution refutation. Otherwise, pop literals off the stack until R is not falsified. Let $S' \leftarrow S' \cup \{R\}$ and go back to step 2.

The procedure halts since at each step, we either increase the size of the stack or deal with a clause. ■

This is actually not too efficient since a large majority of unsatisfiable formulas have resolution refutations exponential in the size of the original formula.

1.2.2 Gentzen's Proof System PK

A **PK proof of a sequent S** for a set of sequents Φ allow sequents from Φ to be leaves in the proof tree. Members of Φ are **non-logical axioms**.

Think of this as a way of reaching an answer of satisfiability or unsatisfiability. Each line in a proof is a **seqent** of the form

$$S = A_1, ..., A_k \rightarrow B_1, ..., B_l$$

Where $\rightarrow$ is *NOT* implication but a new symbol (other texts might use the turnstile $\vdash$). The sequences $A_1, ..., A_k$ and $B_1, ..., B_l$ are **cedents** of the formula S, in particular $A_1, ..., A_k$ is the **antecedent** and $B_1, ..., B_l$ is the **succedent**.

The semantics of sequence is that a t.a. τ satisfies the sequent S iff either τ falsifies some A_i or satisfies some B_i. So S is equivalent to the formula

$$A_S = (A_1 \wedge A_2 \wedge \cdots \wedge A_k) \supset (B_1 \vee B_2 \vee \cdots \vee B_l)$$

where we use $\supset$ to mean implication. If $k = 0$ then

$$A_S = (B_1 \vee B_2 \vee \cdots \vee B_l)$$

and if $l = 0$ then

$$A_S = \neg(A_1 \wedge A_2 \wedge \cdots \wedge A_k)$$

since what we really have is $A_1 \wedge A_2 \wedge \cdots \wedge A_k \supset \emptyset$ which can only be true if some A_i is false. A sequent is **valid** if it is true under all t.a. i.e. the corresponding formula is a tautology.

A formal **proof** in the propositional sequent calculus PK is a finite tree rooted at the**endsequent** (the thing to be prove) with leaves labeled with axioms of the form $A \rightarrow A$.

There are a set of rather long rules which drives the proof forward. Typically we denote finite sequences as Γ, Δ and formulas as A, B. Of particular importance is the cut rule

$\dfrac{\Gamma \rightarrow \Delta, A \quad A, \Gamma \rightarrow \Delta}{\Gamma \rightarrow \Delta}$ The formula A is called the **cut formula**.

Generally a PK proof of formula A is a PK proof of $\rightarrow A$.

Proposition 1.12 *The contraction rules can be derived from the cut rule (with weakening and exchange).*

Proof:

■

Proposition 1.13 *Suppose that we allow $\supset$ (implication) as a primitive connective, rather than one introduced by definition. The left and right introduction rules for $\supset$ is as follows:*

Proof:

■

Theorem 1.14 ***PK Soundness Theorem:*** *every sequent provable in PK is valid.*

Proof: (Intuitively true since each step is valid.) Show that the endsequent in every PK proof is valid by the number of sequents in the proof. Single line proofs are valid since they are just the axioms. To get from one line to the next we use a rule (from the list) so the next line must be valid as well. ■

If a PK proof does not use the cut rule then it is **cut-free**. **Subformula Property:** very formula in every sequent in a cut-free PK proof is a subformula of a formula in the endsequent.

Theorem 1.15 ***PK Completeness Theorem:*** *every valid propositional sequent has a cut free PK proof which does not use the contraction rule.*

Proof: Use the inversion rule. ■

Inversion Principle: for each PK rule except weakening, if the bottom sequent is valid then all top sequents are valid.

Theorem 1.16 ***Derivational Soundness and Completeness:*** *a sequent S is a logical consequent of a set Φ of sequents iff S has a (finite) PK- Φ proof.*

Proof: Prove derivational soundness by induction on the number of sequents in the PK proof. In particular the above soundness theorem show that the validity of the antecedent implies the validity of the succedent while now we will show that the antecedent is a logical consequence of the succedent.

To prove derivational completeness consider a finite set of sequents $\Phi = \{S_1, ..., S_k\}$. Suppose that $\Gamma \to \Delta$ is a logical consequence of $\{S_1, ..., S_k\}$ Using the previous completeness theorem ■

1.2.2.1 Derivational Completeness

Note: a finite proof exists even if Φ is infinite because if $\Phi \vDash S$ then $\Phi_0 \vDash \rho$ for a finite subset Φ_0 of Φ (this uses a variant of the compactness theorem) so it enough to consider a finite set P of formulas.

Proof: First the proof of Derivational Soundness:

$$Q \vDash S \implies S \text{ has a } PK - \Phi - \text{proof}$$

where Q is a set of sequents, S is a sequent. We will use PK-completeness formula A that gives semantics to sequents. Assume ■

Theorem 1.17 ***Compactness Theorem*** *A set Σ of wff (well formed formula) is satisfiable iff every finite subset of it is satisfiable.*

Proof: Enumerate all the wffs: $\alpha_1, \alpha_2, ...$ Proof by contradiction: suppose that a finite subset S of $\Delta = \cup_n \Delta_n$, defined above, is not satisfiable.

Goal: construct a trust assignment τ which satisfies Δ.

Define: $\tau(P) = T \iff P \in \Delta$ for each propositional atom P.

Claim 1.18 $\tau(\phi) = T \iff \phi \in \Delta$ *for any* ϕ*.*

Proof: By structural induction on ϕ. The base case is the proposition symbols by definition of τ assume the claim holds for wffs α and β. Next for the inductive step consider each of the three logical connectives in the standard way. ■

so for (an infinite) Δ, we constructed a truth assignment which satisfies Δ. Since $\Sigma \subseteq \Delta$, so this truth assignment (t.a.) satisfies Σ.

■

Does the proof of compactness in these use this axiom of choice?

1.2.3 Hilbert Style

Observe the following semantic notion:

$$\{A, A \supset B\} \vDash \{B\}$$

rules of inference include: $\dfrac{A, A \supset B}{B}$ these are unconditional.