Pranav Eranki, 3036608969
Adapted from Will T's sheet (with permission)

1 Syntax and Semantics

Syntax

Nonempty set P of **proposition symbols** $\{p_1,\dots,p_n\}$ in a **formal language** $\mathcal{L}(P)$

A set $\{\neg,\wedge,\vee,\rightarrow,\leftrightarrow\}$ of five **connective symbols**

Finite sequence $\langle s_1,\dots,s_n\rangle$ of symbols, an **expression**

Construction sequence for $(\neg p \rightarrow (q\vee r))$: $\langle p,\neg p,q,r,(q\vee r),(\neg p\rightarrow(q\vee r))\rangle$

Truth-Functional Connectives

Unary: 'it is not the case that'; 'the police know that'. Unary connective # is **truth-functional** when, for any proposition p, the truth value of $\#p$ is a *function* of the truth value of p

Binary: 'or'; 'and'; 'if … then'; 'it is more likely that … than it is that'. Binary connective # is **truth-functional** when, for any propositions p and q, the truth value of $p\#q$ is a *function* of those p and q

Soundness, Validity, and Satisfiability

Valid (Tautology): if the premises are true, then the conclusion is also true; ***every** truth table row in which the premises are true, so is the conclusion*

Invalid (Contradiction): true premises but false conclusion; *there is **one** truth table row in which the premise is true while the conclusion is false*

Sound: if valid **and** the premises are true, then the argument is sound

Satisfiable: iff there is *some* valuation that satisfies φ

Semantics

A **valuation** for P is a function V that assigns to each p in P one of the two truth values: $V(p)=0$ or $V(p)=1$

- $\hat V(p)=V(p)$ for each p in P; V satisfies φ if and only if $\hat V(p)=1$. **Notation**: $V\vdash\varphi$
- $\hat V(\neg\varphi)=1-\hat V(\varphi)$
- $\hat V(\varphi\wedge\psi)=\min(\hat V(\varphi),\hat V(\psi))$
- $\hat V(\varphi\vee\psi)=\max(\hat V(\varphi),\hat V(\psi))$
- $\hat V(\varphi\rightarrow\psi)=\begin{cases}1 & \text{if } \hat V(\varphi)\le\hat V(\psi)\\ 0 & \text{otherwise}\end{cases}$
- $\hat V(\varphi\leftrightarrow\psi)=\begin{cases}1 & \text{if } \hat V(\varphi)=\hat V(\psi)\\ 0 & \text{otherwise}\end{cases}$

Valid Forms of Argument

Modus ponens: $\{\varphi\rightarrow\psi,\varphi\}\vdash\psi$

Modus tollens: $\{\varphi\rightarrow\psi,\neg\psi\}\vdash\neg\varphi$

Contraposition: $\{\varphi\rightarrow\psi\}\vdash\neg\psi\rightarrow\neg\varphi$

Disjunctive syllogism: $\{\varphi\vee\psi,\neg\varphi\}\vdash\psi$ and $\{\varphi\vee\psi,\neg\psi\}\vdash\varphi$

Hypothetical syllogism: $\{\varphi\rightarrow\psi,\psi\rightarrow\chi\}\vdash\varphi\rightarrow\chi$

Proof by cases: $\{\varphi\vee\psi,\varphi\rightarrow\chi,\psi\rightarrow\chi\}\vdash\chi$

Examples of Tautologies

- Excluded Middle: $\varphi\vee\neg\varphi$
- $\varphi\rightarrow\varphi$
- $\varphi\rightarrow(\varphi\vee\psi)$, $\psi\rightarrow(\varphi\vee\psi)$
- $(\varphi\wedge\psi)\rightarrow\varphi$, $(\varphi\wedge\psi)\rightarrow\psi$
- $((\varphi\rightarrow\psi)\wedge\varphi)\rightarrow\psi$
- $((\varphi\rightarrow\psi)\wedge\neg\psi)\rightarrow\neg\varphi$
- $((\varphi\rightarrow\psi)\wedge(\psi\rightarrow\chi))\rightarrow(\varphi\rightarrow\chi)$
- $\varphi\rightarrow(\psi\rightarrow\varphi)$
- Pierce's Law: $((\varphi\rightarrow\psi)\rightarrow\varphi)\rightarrow\varphi$

2 Theory and Algorithms

Equivalence

The number of equivalence classes of formulas in $\mathcal{L}(P)$ must be less than the number of n-ary truth-functions.

By the pigeonhole principle, two distinct equivalence classes are never sent to the same n-ary truth function; if this happens, the mapping of the function from equivalence classes of formulas in $\mathcal{L}(P)$ to n-ary truth-functions is **onto**, not one-to-one.

Definition: there are exactly 2^{n^n} equivalence classes of formulas in $\mathcal{L}(\{p_1,\dots,p_n\})$

Every formula in our language is equivalent to a formula in $\mathcal{L}(\{\neg,\wedge\})$, $\mathcal{L}(\{\neg,\vee\})$, and $\mathcal{L}(\{\neg,\rightarrow\})$ **(see section 7 of page 2)**; this is *not* the case for $\mathcal{L}(\{\wedge,\vee,\rightarrow,\leftrightarrow\})$

Truth Functions

Given a **truth function** f, we find a formula that defines it as follows:

1) Find a truth table for f
2) For each row, categorize the true and false propositions and translate into a formula (ex: $p\wedge q\wedge r\wedge\neg t$)
3) Add disjunctions between all conjuncts and negate the entire formula to get $\varphi\equiv f$ (ex: $\varphi=\neg((p\wedge q)\vee(p\wedge\neg q))$)

CNF Algorithm

Note: $(q\leftrightarrow r)$ is equivalent to $\neg((q\rightarrow r)\rightarrow\neg(r\rightarrow q))$. Similarly, $p\rightarrow q$ is equivalent to $\neg p\vee q$

1) Get rid of $\rightarrow$ and $\leftrightarrow$ (see the note)
2) Drive negations in with De Morgan's laws
3) Eliminate double negations
4) Distribute disjunctions over conjunctions

DNF Algorithm

Simply construct a formula based on the rows of the truth table that evaluate to 1

Resolution Algorithm

Used to determine whether φ in CNF are **satisfiable**. Halt the algo in the first step you see an overt contradiction.

1) Find p_1 and $\neg p_1$ in φ
2) Take the disjunction of the literals from the C_{p_1} where p_1 occurs and C_{p_2} where p_2 occurs, except for p_1 and p_2 (ex: resolution of p on $(p\vee q)\wedge(\neg p\vee s)$ gives the resolvent $(q\vee s)$)
3) Take *all* resolvents and add them as conjuncts to the *original formula*
4) If there is no contradiction, then the formula is satisfiable (ex: $(p\wedge\neg p)$ is a contradiction)

Subsumption

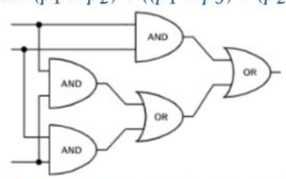
We say that a clause C' is subsumed by a clause C just in case all literals of C are literals of C' (ex: the clause $(p\vee q\vee r)$ is subsumed by the clause $(p\vee q)$)

Combinatorial Problems

1) Encode the constraints into a formula (ex: $\bigwedge_{1\le i\le n}(c_i\wedge\neg m_i\wedge\neg y_i)$)
2) Add conjuncts between those formulas

Circuits with Logic Gates

Ex: $(p_1\wedge p_2)\vee((p_1\wedge p_3)\vee(p_2\wedge p_3))$



3 Monadic Predicate Logic (MPL)

Free/Bound and Open/Closed Formulas

In $\forall xP(x)\rightarrow P(y)$, variable y is **free**, whereas variable x is **bound** by $\forall x$

In $\forall xP(x)\rightarrow P(x)$, variable x in the consequent is **free**, whereas variable x in the antecedent is **bound** by $\forall x$. Variable x is a **free variable**, even though one occurrence is bound.

A formula φ is **open** iff some variable occurs free in φ (ex: $\forall xP(x)\rightarrow P(y)$). A formula is **closed** iff it is not open (ex: $\forall x(P(x)\rightarrow P(x))$)

Formalizing Syllogistic Arguments

Invalid form of syllogism in MPL:

No B is A		$\neg\exists x(B(x)\wedge A(x))$
Some A are C	$\rightarrow$	$\exists x(A(x)\wedge C(x))$
No B is C.		$\neg\exists x(B(x)\wedge C(x))$

Model Syntax and Semantics

A model $\mathcal{M}=(D,I)$ for MPL has domain D and a function I that sends each predicate A to a subset $I(A)\in D$. $I(A)$ is the interpretation of predicate A in $\mathcal{M}$. Ex:

- $Pred=\{\mathtt{Student},\mathtt{Faculty}\}$
- $D=\{1,2,3,4\}$
- $I(\mathtt{Student})=\{1,2\}$
- $I(\mathtt{Faculty})=\{3,4\}$

Validity in Monadic Predicate Logic

An argument with a set Γ of premises and conclusion ψ is **valid** iff: for every model $\mathcal{M}$ and variable assignment g for $\mathcal{M}$, if $\mathcal{M}\vdash_g\varphi$ for each $\varphi\in\Gamma$, then $\mathcal{M}\vdash_g\psi$ (if the model holds true under every variable assignment of all premises, then ψ is a **semantic consequence** of Γ)

Quantifiers

- Universal: $\forall x$; "for all x"
- Existential: $\exists x$; "there exists an x such that"

Examples of Validities

- Duality: $\forall xP(x)\leftrightarrow\neg\exists x\neg P(x)$ and $\exists xP(x)\leftrightarrow\neg\forall x\neg P(x)$
- $\forall$ Instantiation: $\forall xP(x)\rightarrow P(y)$
- $\exists$ Generalization: $P(y)\rightarrow\exists xP(x)$
- Distribution: $\forall x(P(x)\rightarrow Q(x))\rightarrow(\forall xP(x)\rightarrow\forall xQ(x))$
- Vacuous Quantification: $P(x)\rightarrow\forall yP(x)$

Lemma: if a formula φ is *not valid*, then it is falsified in a model on the domain $D=\{1,\dots,2^k\}$ where k is the number of predicate symbols appearing in φ

Constants

In a given model $\mathcal{M}$, the object denoted by a constant is fixed by the model, no matter the variable assignment

Fix a set $Const=\{c_1,c_2,\dots\}$ of **constants** (disjoint from Var and $Pred$)

If $P\in Pred$ and $c\in Const$, then $P(c)$ is a formula (ex: $\mathcal{M}\vdash_g\mathtt{Faculty(Kate)}$ because $I(\mathtt{Kate})=4\in I(\mathtt{Faculty})=\{4\}$; $\mathtt{Kate}$ is a constant)

Note: we need quantifiers only in **infinite** models; in **finite** models, if every object is named by a constant, all quantifiers reduce to conjunction and disjunction

Counting (Identity Predicate)

- At least two: $\exists x_1\exists x_2\ x_1\dot{\neq}x_2$
- At least three: $\exists x_1\exists x_2\exists x_3\ (x_1\dot{\neq}x_2\wedge x_1\dot{\neq}x_3\wedge x_2\dot{\neq}x_3)$
- At most one: $\neg$at least two
- At most two: $\neg$at least three
- Exactly two: at least two $\wedge$ at most two

Validities (Identity Predicate)

- Reflexivity: $t\doteq t$
- Symmetry: $s\doteq t\rightarrow t\doteq s$
- Transitivity: $(s\doteq t\wedge t\doteq u)\rightarrow s\doteq u$

Examples: English to MPL

Let $D(x)$ mean that x is a dog, $C(x)$ mean that x is a cat, and $L(x,y)$ mean that x loves y. Let r refer to Rover. Let s refer to Snuff. Let $f(t)$ be the best friend of t

- Every dog loves a cat: $\forall x(D(x)\rightarrow\exists y(C(y)\wedge L(x,y)))$
- Every cat who loves Rover loves itself: $\forall x((C(x)\wedge L(x,r))\rightarrow L(x,x))$
- Every cat loves its friend: $\forall x(C(x)\rightarrow L(x,f(x)))$
- Rover's friend loves some cat $\exists x(C(x)\wedge L(f(r),x))$
- Someone is loved by Rover $\exists x\,L(r,x)$
- Everyone who loves Rover loves Snuff $\forall x(L(x,r)\rightarrow L(x,s))$

4 Arithmetic

Peano Arithmetic (PA)

(S1) $\forall x\,\neg S(x)\,0$

(S2) $\forall x\forall y\,(S(x)=S(y)\rightarrow x=y)$

(A1) $\forall x\,x+0=x$

(A2) $\forall x\forall y\,x+S(y)=S(x+y)$

(M1) $\forall x\,x\times 0=0$

(M2) $\forall x\forall y\,x\times S(y)=(x\times y)+x$

(IND) $(\varphi^x_0\wedge\forall x(\varphi\rightarrow\varphi^x_{S(x)}))\rightarrow\forall x\varphi$

(E1) $\forall x\,x^0=S(0)$

(E2) $\forall x\forall y\,x^{S(y)}=x^y\times x$

(In slides) $\forall x\,x+1=S(x)$

Theorem (Gödel): PA is negation incomplete

5 Proofs

Standard Concepts

- $Even(x)$: $\exists y\,x=2y$
- $Odd(x)$: $\exists y\,x=2y+1$ or $\exists y(x=S(y+y))$ or $\exists(x=(y\times S(S(0)))+S(0))$
- $x\le y$: $\exists z\,x+z=y$
- $x<y$: $\exists z\,x+S(z)=y$ or $x\le y\wedge\neg y\le x$
- $Prime(x)$: $x\neq 1\wedge\forall y\forall z(x=y\times z\rightarrow(y=1\vee z=1))$

Proof by Induction (Example)

Proof. We proceed by induction on φ.

Base Case ($\varphi=p$): $S(p)=p$ for each proposition symbol in p, so all p translate to p. Therefore, $\varphi\equiv S(\varphi)$.

Inductive Hypothesis: Assume that φ and ψ are equivalent to $S(\varphi)$ and $S(\psi)$ where S sends each formula to a formula in which the only connectives are from $\{\neg,\vee\}$.

Inductive Step: In the cae of negation, $(\neg\varphi)=\neg(\varphi)$, so $\varphi=S(\neg\varphi)$. In the case of conjunction, $\varphi\wedge\psi=\neg(\neg\varphi\vee\neg\psi))$, so $\varphi\wedge\psi\equiv S(\varphi\wedge\psi)$. Therefore, our claim holds by induction.

Complexity

P is the class of problems for which there exists a **polynomial-time algorithm** to solve the problem (# of steps required by algo is bounded above by a polynomial function of the length of the input). **NP** is the class of problems for which there is *no* polynomial-time algorithm

Deterministic algorithms return a *consistent output* for the given input. **Nondeterministic** algorithms return an *inconsistent output* for the given input.

Pranav Eranki, 3036608969
Adapted from Will T's sheet (with permission)

6 Fitch-style Natural Deduction

Soundness and Completeness

Our proof system is (classically) **sound** and **complete**: ψ is a semantic consequence of $\varphi_1,\dots,\varphi_n$ iff there is a proof of ψ from assumptions $\varphi_1,\dots,\varphi_n$

Reiteration

Reiteration says that you may add φ in a new row n if φ occurs above in the main column of the proof or occurs *directly* in a subproof that is still *open* at n

→ Introduction

$\vdots$	$\vdots$	
i	$\quad\alpha$	
$\vdots$	$\quad\vdots$	
j	$\quad\beta$	
$j+1$	$\alpha\to\beta$	$\to$I, i–j

→ Elimination

$\vdots$	$\vdots$	
i	$\alpha\to\beta$	
$\vdots$	$\vdots$	
j	α	
$\vdots$	$\vdots$	
k	β	$\to$E, i, j

∧ Introduction

$\vdots$	$\vdots$	
i	α	
$\vdots$	$\vdots$	
j	β	
$\vdots$	$\vdots$	
k	$\alpha\wedge\beta$	$\wedge$I, i, j

∧ Elimination

$\vdots$	$\vdots$	
i	$\alpha\wedge\beta$	
$\vdots$	$\vdots$	
j	α	$\wedge$E, i
$j+1$	β	$\wedge$E, i

If $\alpha\wedge\beta$, we can independently claim α and β on the same line of the proof

↔ Introduction

To prove $\varphi\leftrightarrow\psi$: assume φ and prove ψ; then assume ψ and prove φ

↔ Elimination

To use $\varphi\leftrightarrow\psi$: Prove φ, then infer ψ; or prove ψ, then infer φ

¬ Introduction

$\vdots$	$\vdots$	
i	$\quad\alpha$	
$\vdots$	$\quad\vdots$	
$j-1$	$\quad\beta\wedge\neg\beta\ (\bot)$	
j	$\neg\alpha$	$\neg$I, i–$j-1$

¬ Elimination

$\vdots$	$\vdots$	
i	$\neg\alpha$	
$\vdots$	$\vdots$	
j	$\alpha\to\beta$	$\neg$E, i

If $\neg\alpha$ appears, you can add $\alpha\to\beta$ for *any* β on the same line of the proof

Ex Falso Quodlibet (EFQ)

$\vdots$	$\vdots$	
i	$\alpha\wedge\neg\alpha$	
$\vdots$	$\vdots$	
j	β	EFQ, i

If $\alpha\wedge\neg\alpha$ appears, you can add *any* β on the same line of the proof

Reductio Ad Absurdum (RAA)

$\vdots$	$\vdots$	
i	$\quad\neg\alpha$	
$\vdots$	$\quad\vdots$	
$j-1$	$\quad\beta\wedge\neg\beta\ (\bot)$	
j	α	RAA, i–$j-1$

∨ Introduction

$\vdots$	$\vdots$	
i	φ	
$\vdots$	$\vdots$	
j	$\varphi\vee\psi$	$\vee$I, i

Given φ, for any ψ, we can claim $\varphi\vee\psi$

∨ Elimination

i	$\alpha\vee\beta$	
$i+1$	$\quad\alpha$	
$\vdots$	$\quad\vdots$	
j	$\quad\varphi$	
$j+1$	$\quad\beta$	
$\vdots$	$\quad\vdots$	
$k-1$	$\quad\varphi$	
k	φ	

On line k: $\vee$E, i, $i+1-j$, $j+1-k-1$

≐ Introduction

$\vdots$	$\vdots$	
n	$t\doteq t$	$=$I

For any term t, you can correctly add $t\doteq t$ on a new line

Transitivity of Identity

1	$a\doteq b$	
2	$b\doteq c$	
3	$a\doteq c$	$=$E, 1, 2

Symmetry of Identity

1	$a\doteq b$	
2	$a\doteq a$	$=$I
3	$b\doteq a$	$=$E, 1, 2

∀ Introduction

$\vdots$	$\vdots$	
i	$\quad\boxed{c}$	
$\vdots$	$\quad\vdots$	
j	$\quad\varphi^x_c$	
$j+1$	$\forall x\varphi$	$\forall$I, i–j

∀ Elimination

i	$\forall x\varphi$	
$\vdots$	$\vdots$	
j	φ^x_t	$\forall$E, i

φ^x_t means t is substitutable for x in φ. Example: $\forall x P(x)$ turns into $P(c)$

∃ Introduction

i	φ^x_t	
$\vdots$	$\vdots$	
j	$\exists x\varphi$	$\exists$I, i

Example: $P(c)$ turns into $\exists x P(x)$

∃ Elimination

$\vdots$	$\vdots$	
i	$\exists x\varphi$	
$i+1$	$\quad\varphi^x_c\ \boxed{c}$	
$\vdots$	$\quad\vdots$	
j	$\quad\psi$	
$j+1$	ψ	

On line $j+1$: $\exists$E, i, $i+1-j$

7 TTs, Translations, and Examples

Truth Tables

P	Q	$P\wedge Q$	$P\vee Q$	$P\to Q$
T	T	T	T	T
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

P	Q	$\neg P$	$P\leftrightarrow Q$
T	T	F	T
T	F	F	F
F	T	T	F
F	F	T	T

Rewriting Formulas Using ¬ and ∧

Proof: Define a function T that sends each formula to a formula as follows:

- $T(p)=p$ for each proposition symbol p;
- $T(\neg\varphi)=\neg T(\varphi)$
- $T(\varphi\wedge\psi)=T(\varphi)\wedge T(\psi)$
- $T(\varphi\vee\psi)=\neg(\neg T(\varphi)\wedge\neg T(\psi))$
- $T(\varphi\to\psi)=\neg(T(\varphi)\wedge\neg T(\psi))$
- $T(\varphi\leftrightarrow\psi)=\neg(T(\varphi)\wedge\neg T(\psi))\wedge\neg(T(\psi)\wedge\neg T(\varphi))$

This is a **recursive definition** of T on the inductively defined set of formulas

Rewriting Formulas Using ¬ and ∨

- $S(p)=p$ for each proposition symbol p;
- $S(\neg\varphi)=\neg S(\varphi)$
- $S(\varphi\wedge\psi)=\neg(\neg S(\varphi)\vee\neg S(\psi))$
- $S(\varphi\vee\psi)=S(\varphi)\vee S(\psi)$
- $S(\varphi\to\psi)=\neg S(\varphi)\vee S(\psi)$
- $S(\varphi\leftrightarrow\psi)=\neg(\neg(\neg S(\varphi)\vee S(\psi))\vee\neg(\neg S(\psi)\vee S(\varphi)))$

Rewriting Formulas Using ¬ and →

For every formula φ, φ is equivalent to $U(\varphi)$, which only contains $\neg$ and $\to$

- $U(p)=p$ for each proposition symbol p
- $U(\neg\varphi)=\neg U(\varphi)$
- $U(\varphi\wedge\psi)=\neg(U(\varphi)\to\neg U(\psi))$
- $U(\varphi\vee\psi)=\neg U(\varphi)\to U(\psi)$
- $U(\varphi\to\psi)=U(\varphi)\to U(\psi)$
- $U(\varphi\leftrightarrow\psi)=\neg((U(\varphi)\to U(\psi))\to\neg(U(\psi)\to U(\varphi)))$

Example 1: ↓ (NOR) Connective

Q: Show how every formula of $\mathcal{L}(P)$ can be translated into an equivalent formula in which the only connective is $\downarrow$ (for 'neither ... nor', or $\neg(t(\varphi)\vee t(\psi))$)

A: $\{\neg,\vee\}$ is a truth-functionally complete set of connectives, so we can translate to formulas containing only connectives in $\{\neg,\vee\}$ to formulas containing only $\downarrow$:

- $t(p)=p$ for each proposition symbol p
- $t(\neg\varphi)=t(\varphi)\downarrow t(\varphi)$
- $t(\varphi\vee\varphi)=(t(\varphi)\downarrow t(\varphi))\downarrow(t(\varphi)\downarrow t(\varphi))$

Example 2: ↑ (NAND) Connective

Q: We extend $\mathcal{L}(P)$ with a new binary connective $\uparrow$ (NAND) so that $\varphi\uparrow\psi$ is equivalent to $\neg(\varphi\wedge\psi)$. Show how every formula of $\mathcal{L}(P)$ can be translated into an equivalent formula in which the only connective is $\uparrow$ (NAND)

A:

- $t(p)=p$ for each proposition symbol p
- $t(\neg\varphi)=t(\varphi)\uparrow t(\varphi)$
- $t(\varphi\vee\psi)=\neg\varphi\uparrow\neg\psi=(\varphi\uparrow\varphi)\uparrow(\psi\uparrow\psi)$