Syntax of PL

- atoms, unary connectives (¬), binary connectives (∧, ∨)

A → B ⊑ (A ∨ B)
A ⇔ B ⊑ (A → B) ∧ (B → A)

The Unit's readability (grammar, for gen. wff is unwraps)

proof assign weights: T: 0 (F: 1), atom p: -1
∧, ∨: 1 ↓

Lemma A formula, weight (formula) = -1
proof structural induction

def syn is a meta language symbol for syntactic equivalence

By lemma: A, B, A', B' formulas and c, c' binary connectives
A c B = syn A' c' B' ⇒ A = A', B = B', c = c'

Semantics of PL

def truth assigns γ: {atoms} → {T, F}
extends to formulas: A^γ = T or A^γ = F

def γ satisfies A iff A^γ = T
γ ——> ⊢ A iff ∀ A ∈ γ, A^γ = T
A, γ is satisfiable iff ∃ γ, A^γ, γ^T = T.
def. $\Phi \vdash A \iff \forall \gamma, \Phi^\gamma = T \Rightarrow A^\gamma = T$

logical (semantic) consequence: $\not\exists U \Phi A$

prop. transitivity of $\vdash$: $\Phi \vdash A \land \Phi \vdash B \Rightarrow \Phi \vdash B$

def. $A$ is valid iff $\emptyset \vdash A$ (also $I \vdash A$), $A$ is a tautology iff

$\forall \gamma, A^\gamma = T$

$A \iff B$ iff $A \iff B \land B \iff A$ (also $A \iff B$)

covers: $P, Q, R$ are distinct atoms, i.e. $P \not\equiv Q, Q \forall P, Q$

$A, B, C$ are arbitrary atoms, i.e. $A \equiv B, \exists A, B$

prop. $\Phi \vdash A \iff \not\exists U \{ \neg A \}$ is unsat.

$A$ is a tautology iff $\neg A$ is unsat.

Thin Duality Theorem

$\forall A$ formula, $A' = [\forall \lambda \exists V] (A) \Rightarrow A' (\iff) \neg A$

proof: structural induction

Thin Craig interpolation theorem

$\forall A, B$ formulas, $S = \{ P \mid PE \land \neg PE \} \neq \emptyset \Rightarrow$ $\exists C$ formula, $\forall PE, C \exists PE, S \land$

$(\emptyset \vdash A \Rightarrow \exists C \forall PE, C \Rightarrow C \Rightarrow B)$

proof: semantic, idea: think about $S = \emptyset$
DNF and CNF

- $A_1 \lor A_2$... disjunc of formulas $A_1$ (same for $\land$)

  $\lor, \land$ are assoc, thus we write $\lor$:  
  
  $(A_1 \lor A_2 \lor A_3)$ instead of $(A_1 \lor (A_2 \lor A_3))$

  $\Delta$ using left assoc!

- def. A **literal** is an abon $P$ or negated abon $\neg P$

- def. A **clause** is a disjunc of literals s.t. none occurs twice.

- A formula is in CNF if it is a conjunc of clauses

  $\lor$ $\neg \phi$ is a CNF formula even though it is not a wff.

- A **$\land$-clause** is a conjunc of

- A **$\lor$-DNF** is a disjunc of $\land$-clauses

  $\land$ $\lor \phi$ is a DNF

The $\lor$ $\phi$ formula, $\phi$ is equivalent (semantic) to a CNF (same $\lor$ $\phi$, $\land$ $\phi$)

$\Delta$ not unique, finding smallest is not in P, tractable

- Formal Proofs

  - Syntax & Motive: don't try all possible $\lor$ (or $\land$) to check validity
Resolution Proof System

Since from defs of unsat and validity, we have:

\( \phi \vdash A \iff \phi \cup \{\neg A\} \) is unsat cons.

\( A \) is Taut iff \( \neg A \) is unsat equiv.

And hence \( \phi \vdash \forall \) everything can be reduced to \( \forall \). Note: Cook approved.

We can show that to establish validity, \( \phi \vdash A \), and unsat, it suffices to show unsat of set of clauses.

The SAT

\[ \exists \text{ polynomial alg.}\; \phi \rightarrow \text{ set}\; \text{s.t.}\; \phi \text{ is unsat iff set is proof } \]

Replace all subformulas in \( \phi \) with new atoms. You get set iff set, but not \( \phi \iff \neg \phi \) since you can assign some random stuff to those guys.

Resolution Rule:

- Consider clauses as sets of literals.

\( C_1 = (A \lor \neg B), C_2 = (\neg A \lor \neg C) \models C_3 = (A \lor \neg B) \)

Soundness: every resolvent is \( \models \) of premises.

\( \text{VC, } C \lor \neg C \text{ dual C } \models \bot \text{ empty clause. unsat!} \)

Def Resolution Refutation: any seq \( C_1 \ldots C_n \) s.t. \( C_n = \bot \) and \( \forall C_i (i \in \mathbb{N}, i \neq n) \), \( C_i \lor C \) on is resolvent of clauses in.

Soundness of Resolution: if \( C_1 \ldots C_n \) is a resol. ref. of \( \phi, \) then \( \phi \) is unsat. / Completeness: converse. see p.8-9.
Computability

Register Machines

Input: Computes a (partial) function over \( \mathbb{N} \).

Def: \( f : (\mathbb{N} \cup \{0\})^n \to \mathbb{N} \cup \{0\}; \quad (n \geq 0) \)

\[ f(c_1, \ldots, c_n) = \infty \iff \exists c_i, c_i = \infty \]

Def: domain (\( f \)) = \{ \( \bar{x} \in \mathbb{N}^m \mid f(\bar{x}) \neq \infty \) \}

Def: \( f \) is total iff domain (\( f \)) = \( \mathbb{N}^m \)

Function taxonomy:
\[ f : X \to Y \]
\[ x \mapsto f(x) = y \]
\[ x \in X \quad y \in Y \]

(partial) func^0
Def: \( \forall x, \exists! y, f(x) = y \vee f(x) = \infty \)

Def: \( \forall x, \exists! y, f(x) = y \)

Def: \( \forall y, \exists x, f(x) = y \)

Def: \( \forall x, \exists! y, f(x_1) \neq f(x_2) \) \( \ast > x \) \( \ast \) if \( f \) is partial; otherwise \( f(x_1) = \infty \) or \( f(x_2) = \infty \)

(total) func^0
Def: \( \forall x, \exists! y, f(x) = y \land \forall y, \exists! x, f(x) = y \)

In fact, if \( f \) is injective, then \( f(x) = \infty \) or \( f(x) = \infty \)

Def: \( \forall x, \exists! y, f(x) = y \) \( \ast \) \( \forall y, \exists! x, f(x) = y \)
Thus the input to \( P \) running on an RM will be the arguments to the partial function \( P \) is computing; each arg is stored in corresponding registers (i.e. \( f(a_1, ..., a_m) \Rightarrow R_i = a_i \))

- **Output:** \( R_1 \) contains \( f(a_1, ..., a_m) \) when \( P \) halts.
- **States:** If \( P \) doesn't halt, \( f(a_1, ..., a_m) = \infty \).

**Def:** a state is an \((m+1)\)-tuple \( < K, R_1, ..., R_m > \). Given a state \( s \), the next state \( Nextp(s) \) is the state resulting when command \( c_k \) is applied.

**Def:** a halting state is a state such that \( K = h \); in which case \( Nextp(\text{shall}) = \text{shall} \).

**Primitive Recursion:**

\( f \) is P.R. iff \( f \) can be obtained from initial functions:

\[
\begin{align*}
T & : \text{ 0-ary func. equal to 0} \\
S & : \text{ 1-ary successor func. } S(x) = x + 1 \\
I_{i,j} & : i(x_1, ..., x_n) = x_i : \text{ Infinite class of project func.}
\end{align*}
\]

by composition:

\[ f(\overline{x}) = g(h_1(\overline{x}), ..., h_m(\overline{x})) \]

or by primitive recursion:

\[
\begin{align*}
& f(\overline{x}, 0) = g(\overline{x}) \\
& f(\overline{x}, y+1) = h(\overline{x}, y, f(\overline{x}, y))
\end{align*}
\]
Recursive  &  Recursively Enumerable Sets

relational \( R \subseteq \mathbb{N}^m \) is an \( m \)-ary predicate

is a set

decidable if is a total boolean function \( A(x) = \begin{cases} 0 & \text{if } x \notin \mathbb{N}^m \\ 1 & \text{otherwise} \end{cases} \)

Def: \( R \) is rec iff \( A(x) \) is computable -

S-m-n Theorem

\( \forall m, n \geq 0, \exists \) total computable \((m+1)\)-ary function \( S^m_n \) s.t.

\( \{ S^m_n (z, \bar{y}) \}^m_n(x) = \{ z \}^m_n (\bar{y}, \bar{x}) \)

Kleene Theorem

\( \forall m \geq 1, \) the \((m+2)\)-ary relation \( T_m(z, \bar{x}, y) \) where \( y \) codes the computation of \( \{ z \}^m_n (\bar{x}) \) is primitive recursive -

KNF Theorem

output function \( U(y) = \)

\( \forall m \geq 1, \exists \) primitive recursive \( U \land T_m \) s.t.

\( \{ z \}^m_n (\bar{x}) = U(\mu y. T_m(z, \bar{x}, y)) \)
Completeness

- Arithmetical relations: representable (i.e. \( R(\bar{a}) \subseteq \mathbb{N} = A(\bar{a}) \))
- \( \Delta_0 \) relations: bounded formula translated from \( \Delta_1 \subseteq \Delta_0 \) formula
- \( \Delta_0 \) relations \( \preceq \) PR
- \( \exists \Delta_0 \) relation: representable by \( \exists \Delta_0 \) formula \( \subseteq \Sigma_1 \) formula
- \( \exists \Delta_0 \) relations \( \preceq \) RE relations \((\preceq \exists \Delta_1 \text{ known})\)

Hard Lemma: if \( f \in \text{PR} \) then graph \((f) = R(\bar{x}, y) = (y = f(\bar{x}) \) is RE.

Proof: \( B(\bar{c}, 0, i) \)

Wild notes

\[ \text{PA U \{ \neg \text{con}(PA) \}} \] is consistent extension of \( \text{PA} \)

Every consistent \( \Sigma \)

\[ P_1: \forall x \ (sx \neq 0) \]

\[ P_2: \]

\[ P_3: \]

\[ \text{UNDECIDABILITY Theorem: } \]

\[ \exists \text{ represent RE rel.} \]

\[ \text{RE IDA} \]

\[ \text{COMP \land axio = decid.} \]

\[ \text{PA is RE} \]

\[ \exists A \text{ is RE } \leftrightarrow \exists y \ R(x, y); \text{if } \]

\[ A \subseteq \bar{x} \subseteq A \]

\[ A = \text{dom}(f) \text{ where } f(x) = y \iff R(x, y) \]

\[ = \{ \text{for } R \text{ be } R \text{ E}\}

\[ \text{by K-NF} \]

\[ \text{p28} \]

\[ g(x, y) = f(x), (y) \]

Then \( B \) s. w. any total comp

\[ g(x, y) = f(f(x)), (y) \]