CSC438

. Syntax of PL

- atoms, unary connective ($\neg$), binary connectives ($\wedge, \vee$)

$A \supset B \iff (\neg A \vee B)$

$A \leftrightarrow B \iff (A \supset B) \wedge (B \supset A)$

Thm Unique readability (grammar for gen. wff is unamb.)

proof assign weights : $\neg : 0$ | $(: 1$ | atom $P : -1$

$\wedge, \vee : 1$ | $) : -1$ |

Lemma $\forall$ formula, weight(formula) $= -1$

proof struct. induc.

def $=_{syn}$ is a meta language symbol for syntactic equivalence.

By Lemma : A, B, A', B' formulas and c, c' binary connectives

$$A c B =_{syn} A' c' B' \Rightarrow A = A', B = B', c = c'$$

. Semantics of PL

def . truth assignt $\tau : \{atoms\} \longrightarrow \{T, F\}$

. extends to formulas : $A^\tau = T$ or $A^\tau = F$

def . τ satisfies A (formula) iff ~~...~~ $A^\tau = T$

. τ —— Φ (set) iff $\forall A \in \Phi, A^\tau = T$

. A, Φ is satisfiable iff $\exists \tau, A^\tau, \Phi^\tau = T$.

def $\Phi \models A$ iff $\forall \tau,\ \Phi^{\tau} = T \Rightarrow A^{\tau} = T$

logical (semantic) consequence

prop transitivity of $\models$: $\Phi \models A \wedge \Phi \cup \{A\} \models B \Rightarrow \Phi \models B$

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

iff $\forall \tau,\ A^{\tau} = T$

. $A \Leftrightarrow B$ iff $A \models B \wedge B \models A$ (also $A \mathrel{=\!\mid\!\models} B$)

conven: P, Q, R are distinct atoms, i.e. $P \neq_{syn} Q \quad \forall P, Q$

A, B, C are arbitrary atoms, i.e. $A =_{syn} B \quad \exists A, B$

prop $\Phi \models A$ iff $\Phi \cup \{\neg A\}$ is unsat.

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

Thm Duality theorem

$$\forall A \text{ formula},\ A' = [\wedge \mapsto \vee,\ \vee \mapsto \wedge,\ P \mapsto \neg P](A) \Rightarrow A' \Leftrightarrow \neg A$$

proof structural induction

Thm Craig interpolation theorem

$$\forall A, B \text{ formulas},\ S = \{P \mid P \in A \wedge P \in B\} \neq \emptyset \Rightarrow$$

change obsolete notation! no sets

$$\left[\emptyset \models A \supset B \Rightarrow \exists C\ \overset{\text{interpolant}}{\text{formula}},\ (\forall P \in C,\ P \in S) \wedge (\emptyset \models A \supset C \wedge \emptyset \models C \supset B)\right]$$

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

. DNF and CNF

def. $A_1 \vee A_2 \ldots$ disjunc° of formulas A_i (same for $\wedge$)

conv $\vee, \wedge$ are assoc, thus we write wlog:

$(A_1 \vee A_2 \vee A_3)$ instead of $((A_1 \vee A_2) \vee A_3)$

⚠ using left assoc!

def A literal is an atom P or negated atom $\neg P$

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

CNF

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

  conv $\wedge \emptyset$ is a CNF formula even though it is not a wff.

DNF

- A $\wedge$-clause is a conjunc° of ———
- A ——— DNF ——— disjunc° of $\wedge$-clauses

  conv $\vee \emptyset$ is a DNF ———

Thm $\forall A$ formula, A is equivalent (semantic) to a CNF (same for DNF)

⚠ Not unique, finding smallest is not in P. (tractable)

. Formal Proofs

Syntactic. Motiva°: don't try all possible τ (2^n for n atoms) to check validity

→ Resolution Proof System

. Since from defs of unsat and validity, we have: $\begin{cases} \Phi \models A \text{ iff } \Phi \cup \{\neg A\} \text{ is unsat} & \text{cons.} \\ A \text{ is TAUT iff } \neg A \text{ is unsat} & \text{equiv.} \end{cases}$

Note: ask Cook: derivable

material / syntactic	logical / semantic	~~syntactic~~ provable
~~...~~ $\supset$	$\models$, $\Rightarrow$	$\vdash$
$=_{syn}$	$\dashv\models$, $\Leftrightarrow$	$\dashv\vdash$

Note: Cook approved.

. And ~~...~~ everything can be reduced to CNF. We ~~...~~ can show that to establish validity, $\Phi \models A$, and unsat, it suffices to show unsat of set of clauses.

Thm SAT

$\exists$ polytime alg: ~~...~~ $\Phi \mapsto S_\Phi$ such that Φ is sat iff S is sat.

proof p.7 → replace all subformulas in Φ with new atoms. You get sat iff sat; but not $\Phi \Leftrightarrow S_\Phi$ since you can assign some random stuff to those guys.

. Resolution Rule:

→ Consider clauses as sets of literals.

→ $C_1 = (A \vee \ell), C_2 = (B \vee \bar{\ell}) \models C_3 = (A \vee B)$ resolvant

Soundness principle: every resolvant is $\models$ of premises.

$\forall C$, C, ~~...~~ dual $C \models \vee\emptyset$ ← empty clause. unsat!

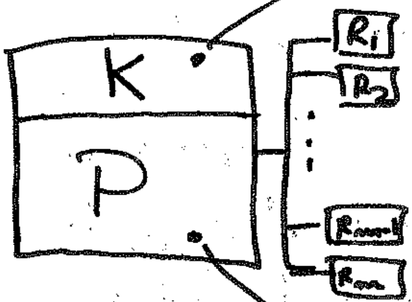
Def Resolution refutation for S: any seq $C_1 .. C_q$ s.t. $C_q = \vee\emptyset$ and $\forall C_i$ ($i \in \mathbb{N}, i \neq q$), $C_i \in S$ or is resolvant of clauses in S.

• Soundness of Resolution: if $C_1 .. C_q$ is a resol. refu. of S, then S is unsat. / Completeness: converse. see p.8-9.

Computability

· Register Machines

instruction counter/pointer $K \in \{0..h\}$



m Registers, each stores an integer.
Output register: R_1

3 commands and Encoding:

com.	enc.
$R_i \leftarrow 0$	Z_i $(i \in \mathbb{N})$
$R_i \leftarrow R_i + 1$	S_i $(i \in \mathbb{N})$
goto k if $R_i = R_j$	$J_{i,j,k}$ $\begin{cases} (i,j \in \mathbb{N}) \\ 0 \le k \le h \end{cases}$

$P = \langle c_0 .. c_{h-1} \rangle$ ← program: sequence of commands (encoded as follows)

com. is h halting

→ ~~State~~ Input: ~~P~~ computes a (partial) function f over $\mathbb{N}$.

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

$$f(c_1, .. c_n) = \infty \Leftarrow \exists c_i, c_i = \infty$$

Def: $\text{domain}(f) = \{ \vec{x} \in \mathbb{N}^n \mid f(\vec{x}) \neq \infty \}$

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

---

function taxonomy: $f : X \to Y$, $x \mapsto f(x) = y$ $\quad \Big| \; x \in X, \; y \in Y$

(partial) func°	$\forall x, \exists! y, f(x) = y \vee f(x) = \infty$
total func°	$\forall x, \exists! y, f(x) = y$
surjective p. (onto)	$\forall y, \exists x, f(x) = y$
injective p. (one-to-one)	$\forall x_1, x_2, f(x_1) \neq f(x_2)$ * or if f is partial: $f(x_1) = \infty$ or $f(x_2) = \infty$
total bijection	$\underbrace{\forall x, \exists! y, f(x) = y}_{\text{totality}} \wedge \underbrace{\forall y, \exists! x, f(x) = y}_{\text{(sur)+inj)ectivity}}$

all others regs are set to 0 - | Thus the input to P running as an RM will be the arguments to the partial func° P is computing; each arg is stored in corresponding registers (i.e. $f(a_1 .. a_n) \Rightarrow R_1 = a_1$)

→ Output: R_1 contains $f(a_1 .. a_n)$ when P halts.
If P doesn't halt, $f(a_1 .. a_n) = \infty$ -

→ States:

Def: a state is an $(m+1)$-tuple $\langle K, R_1 .. R_m \rangle$

Given a state s, the next state $Next_P(s)$ is the state resulting when command c_K is applied -

Def: a halting state s_{halt} is a state such that $K = h$; in which case $Next_P(s_{halt}) = s_{halt}$ -

. Primitive Recursion:

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

$$\begin{cases} Z : & 0\text{-ary func. equal to } 0 \\ S : & 1\text{-ary successor func. } S(x) = x+1 \\ I_{n,i}(x_1 .. x_n) = x_i : & \text{Infinite class of projec° func.} \end{cases}$$

by composition:

$$f(\vec{x}) = g\left(h_1(\vec{x}) .. h_m(\vec{x})\right)$$

or by primitive recursion:

$$\begin{cases} f(\vec{x}, 0) = g(\vec{x}) \\ f(\vec{x}, y+1) = h(\vec{x}, y, f(\vec{x}, y)) \end{cases}$$

Recursive & Recursively Enumerable Sets

relation $R \subseteq \mathbb{N}^n$ $\begin{cases} \text{is an } n\text{-ary predicate} \\ \text{is a set} \\ \text{is a total boolean function } A(\vec{x}) = \begin{cases} 0 & \text{if } \vec{x} \in A \\ 1 & \text{o/wise} \end{cases} \end{cases}$

decidable

Def R is rec iff $A(\vec{x})$ is computable.

S-m-n Theorem

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

$$\{S^m_n(z, \vec{y}^{\,m})\}_n(\vec{x}^{\,n}) = \{z\}_{m+n}(\vec{y}^{\,m}, \vec{x}^{\,n})$$

~~KNF~~ Theorem (Kleene ~~Theorem~~)

$\forall n \geq 1$, the $(n+2)$-ary relation $T_n(z, \vec{x}^{\,n}, y)$ where y codes the computation of $\{z\}_n(\vec{x}^{\,n})$ is primitive recursive.

KNF Theorem

output function $- U(y) =$

$\forall n \geq 1$, $\exists$ primitive recursive U $\wedge$ $\exists T_n$ s.t.

$$\{z\}_n(\vec{x}^{\,n}) = U(\mu y \, T_{n+2}(z, \vec{x}^{\,n}, y))$$

Σ completeness

- Arithmetical relation: representable (i.e. $R(\vec{a}) \leftrightarrow \underline{\mathbb{N}} \models A(s_{\vec{a}})$) by A on $\mathcal{L}_A$
- Δ_0 relation: representable by bounded formula translated from $\mathcal{L}_{A,\leq}$ (Σ_0 formula)
  - Δ_0 relations $\subsetneq$ PR
- $\exists\Delta_0$ relation: representable by $\exists\Delta_0$ formula $\in \Sigma_1$ formula
  - $\exists\Delta_0$ relations $\subseteq$ RE relations ($\Leftarrow \exists\Delta$ theorem)
- Main Lemma: if $f \in PR$ then graph(f): $R(\vec{x}^n, y) = (y = f(\vec{x}^n))$ is RE.

  proof: $\beta(c, d, i)$

Wild notes

$PA \cup \{\neg con(PA)\}$ is consistent extension of PA

predicate

Every consistent Σ

$P1: \forall x\,(sx \neq 0)$

$P2$:

$P3$:

$P4$: +

$P5$: *

$P6$:

UNDECIDABILITY Theorem: Σ represent RE rela·

DECIDA

comp $\wedge$ axio $\Rightarrow$ decid.

A is RE $\leftrightarrow \exists y\, R(x,y)$ iff $x \in A$

$A = dom(f)$ where $f(x) = \mu y\, R(x,y)$

$\Rightarrow f \in R$ bc $R \in R$

By KNFT

p28

$g(x,y) = \{x\}_1(y)$

Then $\exists f$ unary total comp

$g(x,y) = \{f(x)\}_1(y)$

PA is RE