Satisfiability

Victor W. Marek

Computer Science

University of Kentucky

Spring Semester 2005

Satisfiability story

- Satisfiability - invented in the 20^{ies} of XX century by philosophers and mathematicians (Wittgenstein, Tarski)
- Shannon (late 20^{ies}) applications to what was then known as electrical engineering
- Fundamental developments: 60^{ies} and 70^{ies} – both mathematics of it, and fundamental algorithms
- 90^{ies} - progress in computing speed and solving moderately large problems
- Emergence of “killer applications” in Computer Engineering, Bounded Model Checking

Current situation

- SAT solvers as a class of software
- Solving large cases generated by industrial applications
- Vibrant area of research both in Computer Science and in Computer Engineering
  - Various CS meetings (SAT, AAAI, CP)
  - Various CE meetings (CAV, FMCAD, DATE)
  - CE meetings Stressing applications

This Course

- Providing mathematical and computer science foundations of SAT
  - General mathematical foundations
  - Two- and Three- valued logics
  - Complete sets of functors
  - Normal forms (including *ite*)
  - Compactness of propositional logic
  - Resolution rule, completeness of resolution, semantical resolution
  - Fundamental algorithms for SAT
  - Craig Lemma

And if there is enough of time and will...

- Easy cases of SAT
  - Horn
  - 2SAT
  - Linear formulas
- And if there is time (but notes will be provided anyway)
  - Expressing runs of polynomial-time NDTM as SAT, NP completeness
  - “Mix and match”
  - Learning in SAT, partial closure under resolution
  - Bounded Model Checking

Various remarks

- Expected involvement of Dr. Truszczynski
- An extensive set of notes (> 200 pages), covering most of topics will be provided f.o.c. to registered students (in installments)
- No claim to absolute correctness made
- Syllabus: http://www.cs.uky.edu/ marek/htmldid.dir/686.html
- Homeworks every other week
- Midterm
- Individually-negotiated project
  - Write your own SAT solver [a]
  - Modify Chaff
  - Use SAT solver for some reasonable task
- Questions?

[a] Hic Rhodes, Hic Salta

Basic concepts

- Sets and operations on sets
- Relations
- Partial orderings (posets)
- Elements' classification
- Lattices
- Boolean Algebras
- Chains in posets, Zorn Lemma
- Well-orderings, ordinals, induction
- Inductive proofs
- Inductive definability

Fixpoint theorem

- Complete Lattices
- Monotone operators (functions) in lattices
- (**Knaster-Tarski Fixpoint Theorem**) If L is a complete lattice and $f : L \to L$ is monotone operator in L then f possesses a fixpoint. In fact fixpoints of f form a complete lattice under the ordering of L, and thus there is a least and largest fixpoint of f
- Continuous monotone functions
- The least fixpoint (but *not* the largest fixpoint) of a continuous operator reached in ω or less steps
- Generalizations of fixpoint theorem

Syntax of propositional logic

- Variables of some (problem-dependent) set *Var*
- For each set *Var* a separate propositional logic
- Inductive definition of the set of formulas $Form_{Var}$
- Thinking about formulas as binary trees, the rank of formula

Semantics

- Valuations of variables
- Partial valuations of variables
- Two-valued logic and valuations
- Type *Bool*
- Tables for operations in *Bool*
- Valuations acting on formulas
- Valuations *uniquely* extend from variables to formulas
- Characterizing valuations as complete sets of literals
- Characterizing valuations as sets of variables
- Two-valued truth function v_2
- Satisfaction relation $\models$
- Consistent theories

Localization and joint-consistency

- Interactions between sets of variables
- Restrictions of valuations
- **Localization theorem**: If $Var_1 \subseteq Var_2$ and $\varphi \in \mathcal{L}_{Var_1}$, v is a valuation of Var_2, v' is the restriction of v to $Var_1 \models \varphi$ then $v \models \varphi$ if and only if $v' \models \varphi$
- Complete sets of formulas
- **Lemma**: Complete sets of formulas determine valuations and conversely, valuations determine completes sets of formulas
- (Robinson joint-consistency) Let T_1, T_2 are two sets of formulas in Var_1, Var_2 resp. Let us assume that $Var = Var_1 \cap Var_2$ and that both T_1, T_2 are complete for Var and coincide. Then $T_1 \cup T_2$ is consistent

Partial valuations, 3-valued logic

- Need for partial valuations?
- Partial valuation as complete valuations but in $\{0, 1, u\}$
- Post ordering and Kleene ordering in the set $\{0, 1, u\}$
- Product ordering
- Post ordering and Kleene ordering in the multiple-copies product of $\{0, 1, u\}$, $\leq_p$ and $\leq_k$
- Getting Kleene and Post orderings of partial valuations
- Valuations as maximal partial valuations

Tables for Kleene 3-valued logic

- Three-valued truth function v_3
- v_2 and v_3 coincide if v is a (two-valued) valuation
- If $v \leq_k w$ then for every formula φ, $v_3(\varphi) \leq_k w_3(\varphi)$
- Autarkies
- Fundamental effect of this result in SAT
- Restriction result for 3-valued valuations

Tautologies and Satisfiability

- A *tautology* - formula true under all valuation of its variables
- Satisfiable formulas
- A formula φ is satisfiable if and only if $\neg\varphi$ is not a tautology
- Consequences of this fact: satisfiability checkers as tautology checkers
- Common tautologies
- How many are there tautologies?

Substitutions to formulas

- Substitution

$$\begin{pmatrix} p_1 & \dots & p_n \\ \psi_1 & \dots & \psi_n \end{pmatrix}$$

- Valuations acting on substitutions
- **Substitution Lemma:** Let $\varphi \in Form_{\{p_1,\dots,p_m\}}$ be a formula in propositional variables $p_1, \dots, p_m$ and let $\langle \psi_1, \dots, \psi_m \rangle$ be a sequence of propositional formulas. Let v be a valuation of all variables occurring in $\psi_1, \dots, \psi_m$. Finally, let v' be a valuation of variables $p_1, \dots, p_m$ defined by $v'(p_j) = v(\psi_j)$, $1 \leq j \leq m$. Then

$$v'(\varphi) = v\left(\varphi \begin{pmatrix} p_1 & \dots & p_n \\ \psi_1 & \dots & \psi_n \end{pmatrix}\right).$$

Substitutions to formulas, cont'd

- Let φ be a tautology, with variables of φ among $p_1, \ldots, p_n$. Then for every choice of formulas $\langle \psi_1, \ldots, \psi_n \rangle$, the formula $\varphi \begin{pmatrix} p_1 & \ldots & p_n \\ \psi_1 & \ldots & \psi_n \end{pmatrix}$ is a tautology.
- There are infinitely many tautologies

Lindenbaum Algebra

- Relation $\sim$: $\varphi \sim \psi$ if for all valuations v (of all variables occurring in φ, ψ) $v(\varphi) = v(\psi)$
- $\sim$ is an equivalence relation
- $\varphi \sim \psi$ iff the formula $\varphi \equiv \psi$ is a tautology
- We can form the cosets $Form/\sim$
- Operations in $Form/\sim$
- Independence from the choice of representatives
- Lindenbaum Algebra
- Lindenbaum Algebra is a Boolean Algebra

Permutations of atoms and literals

- Permutations of atoms
- Permutations of literals, consistent permutations of literals
- Shifts
- Consistent permutations form a group
- Decomposing consistent permutations of literals into shifts and permutation of variables
- There is $2^n \cdot n!$ of consistent permutations of literals over a set Var of size n
- Consistent permutations of literals (and thus permutations of variables) preserve completeness of sets of literals
- Every consistent and complete set of literals can be mapped onto any other consistent and complete set of literals by a suitably chosen consistent permutation of literals

Permutations and formulas

- Permutations act on formulas
- (**Permutation Theorem**) If φ is a formula, v is a valuation and π a consistent permutation of literals then $v \models \varphi$ if and only if $\pi(v) \models \pi(\varphi)$
- Set-representation of valuations and permutation

Semantical consequence

- Operation *Cn*, entailment of the (sets of) formulas

$$Cn(F) = \{\varphi : \forall_v (v \models F \Rightarrow v \models \varphi)\}$$

- Cn is an operator in the complete Boolean algebra of sets of formulas
- For all sets F, $F \subseteq Cn(F)$
- Cn is monotone and idempotent
- Cn is continuous (but we have no means to prove it, yet)
- $Cn(\emptyset)$ consists of tautologies and nothing else

Deduction Theorem

- Implication functor and consequence operation
- The following are equivalent
  - $\varphi \Rightarrow \vartheta \in Cn(F)$
  - $\vartheta \in Cn(F \cup \{\varphi\})$

Operations *Mod* and *Th*

- From sets of formulas to valuations: $Mod(F) = \{v : v \models F\}$
- From sets of valuations to formulas $Th(V) = \{\varphi : \text{ for all } v \in V, v \models \varphi\}$
- Connection: Let v be a valuation and V a set of valuations. Then $v \in Mod(Th(V))$ if and only if for every finite set of variables A there is a valuation $w \in V$ such that $v\mid_A = w\mid_A$.
- Let A be a finite set of propositional variables. Let V be a collections of valuations of set A and v be a valuation of A. Then $v \models Mod(Th(V))$ if and only if $v \in V$.
- $Mod(Th(Mod(F))) = Mod(Cn(F)) = Mod(F)$
- $Th(Mod(Th(V))) = Th(V)$.

Functors and formulas

- An n-ary functor is a n-ary function in $Bool$
- There are 2^{2^n} n-ary functors
- Table - Boolean function of finite number of variables
- Altogether there is infinitely many tables
- Given a set $\mathcal{C}$ of names for functors we can form $Form^{\mathcal{C}}$ - the set of formulas based on $\mathcal{C}$ (obvious syntactic restrictions)

Completeness of sets of functors

- Valuations as before, functions on Var into *Bool*
- Assigning tables to well-formed formulas (i.e. trees, but no longer binary trees)
- T_φ table associated with φ
- $\mathcal{C}$ is *complete* if every table T is equal to T_φ for some $\varphi \in Form^{\mathcal{C}}$

Completeness of sets of functors, cont'd

- If $\mathcal{C}, \mathcal{C}_1$ are two sets of functors, $\mathcal{C}$ is complete and $\mathcal{C} \subseteq \mathcal{C}_1$ then $\mathcal{C}_1$ is also complete.
- The set $\{\neg, \wedge, \vee\}$ is a complete set of functors.
- Thus $\{\neg, \wedge, \vee, \Rightarrow, \equiv\}$ is a complete set of functors.
- Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two sets of functors. Let us assume that $\mathcal{C}_2$ is a complete set of functors, and that for every functor $c \in \mathcal{C}_2$ there is a formula $\varphi \in Form^{\mathcal{C}_1}$ such that the table T_c is identical with the table T_φ. Then $\mathcal{C}_1$ is also a complete set of functors.
- Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two sets of functors. Let us assume that $\mathcal{C}_2$ is a complete set of functors, and that for every functor $c \in \mathcal{C}_2$ there is a formula $\varphi \in Form^{\mathcal{C}_1}$ such that the formula $c(x_1, \ldots, x_n) \equiv \varphi(x_1, \ldots, x_n)$ is a tautology. Then $\mathcal{C}_1$ is a complete set of functors.

Completeness of sets of functors, cont'd

- $\{\neg, \wedge\}$ is a complete set of functors
- $\{\neg, \vee\}$ is a complete set of functors
- $\{\Rightarrow\}$ is a complete set of functors.
- But in this last case we allowed use of $\perp$
- $\{ite\}$ is complete, but use of constants needed
- Beautiful property of *ite*:
  Let φ be a propositional formula and let x be a variable. Then for all variables y and formulas ψ and ϑ the equivalence

$$\varphi\binom{x}{ite(y, \psi, \vartheta)} \equiv ite(y, \varphi\binom{x}{\psi}, \varphi\binom{x}{\vartheta})$$

  is a tautology.

More on completeness

- The set $\{\wedge, \vee\}$ is not a complete set of functors, even in the weak sense.
- The set $\{\equiv\}$ is not complete

Normal Forms, starting with negation

- Negation normal form, pushing negation downwards
- Double negation rewrite rule
- De Morgan laws as rewrite rules
- Canonical negation normal form

Occurrences of variables

- Inductive definition of positive and negative occurrences
- A variable may occur both positively and negatively in a formula
- Canonical negative normal form preserves both positive and negative occurrences of variables

Alternative way of getting the occurrences result

- Treating formulas as trees
- Counting negations on the unique path from the leaf to the root
- Positive occurrrence if this number is even, odd occurrence o/w
- Rewrite rules, when interpreted as operations on trees preserve even and odd number of occurrences on paths
- In $\mathrm{cNN}(\varphi)$ number of occurrences 0 or 1

cDNF, cCNF

- Canonical disjunctive and conjunctive normal forms:
- Four distributive rules
  - $\varphi \wedge (\psi \vee \vartheta) \equiv (\varphi \wedge \psi) \vee (\varphi \wedge \vartheta)$
  - $(\psi \vee \vartheta) \wedge \varphi \equiv (\psi \wedge \varphi) \vee (\vartheta \wedge \varphi)$
  - $\varphi \vee (\psi \wedge \vartheta) \equiv (\varphi \vee \psi) \wedge (\varphi \vee \vartheta)$
  - $(\psi \wedge \vartheta) \vee \varphi \equiv (\psi \vee \varphi) \wedge (\vartheta \vee \varphi)$

Handling cDNF

- Formula φ is already in cNN form
  - $\text{cDNF}(a) = a$ if a is a variable
  - $\text{cDNF}(\neg a) = \neg a$ if a is a variable
  - Assuming $D_1^1 \vee D_2^1 \vee \ldots \vee D_{m_1}^1 = \text{cDNF}(\psi_1)$ and $D_1^2 \vee D_2^2 \vee \ldots \vee D_{m_2}^2 = \text{cDNF}(\psi_2)$, define
    1. $\text{cDNF}(\psi_1 \wedge \psi_2) = \bigvee_{i \leq m_1, j \leq m_2} D_i^1 \wedge \text{D}_j^2$
    2. $\text{cDNF}(\psi_1 \vee \psi_2) = \text{cDNF}(\psi_1) \vee \text{cDNF}(\psi_2)$
- For every formula φ in negation normal form, the formula $\text{cDNF}(\varphi)$ is in disjunctive normal form. Moreover, $\varphi \equiv \text{cDNF}(\varphi)$.
- (Preparing for the complete DNF).
  For every formula φ and a variable $p \in \mathit{Var}_\varphi$ there exist two formulas ψ_i, $i = 1, 2$ such that
  1. $p \notin \mathit{Var}_{\psi_i}$, $i = 1, 2$
  2. The formula $\varphi \equiv ((p \wedge \psi_1) \vee (\neg p \wedge \psi_2))$ is a tautology.

Conjunctive Normal Form

- Assuming φ in cNN form
- $\mathrm{cCNF}(a) = a$ if a is a variable
- $\mathrm{cCNF}(\neg a) = \neg a$ if a is a variable
- Assuming $C_1^1 \wedge C_2^1 \wedge \ldots \wedge C_{m_1}^1 = \mathrm{cCNF}(\psi_1)$ and $C_1^2 \wedge C_2^2 \wedge \ldots \wedge C_{m_2}^2 = \mathrm{cCNF}(\psi_2)$, define
  1. $\mathrm{cCNF}(\psi_1 \vee \psi_2) = \bigwedge_{i \leq m_1, j \leq m_2} C_i^1 \vee \mathrm{C}_j^2$
  2. $\mathrm{cCNF}(\psi_1 \wedge \psi_2) = \mathrm{cCNF}(\psi_1) \wedge \mathrm{cCNF}(\psi_2)$
- For every formula φ in negation normal form, the formula $\mathrm{cCNF}(\varphi)$ is in conjunctive normal form. Moreover, $\varphi \equiv \mathrm{cCNF}(\varphi)$.

What if only positive (negative) occurrences?

- If a variable p has only positive occurences in F, $\{p\}$ does not have to be an autarky, but...
- Let F be a set of formulas in which a variable p has only positive (resp. negative) occurrences. Then F is satisfiable if and only if $F \cup \{p\}$ (resp. $F \cup \{\neg p\}$) is satisfiable. In other words, F is satisfiable if and only if it is satisfiable by a valuation v such that $v(p) = 1$ (resp. $v(p) = 0$).
- Look at the proof: it shows equivalent formulas may not have same autarkies

Reduced Normal Forms

- A clause $C = p_1 \vee \ldots \vee p_k \vee \neg q_1 \vee \ldots \vee \neg q_l$ is a tautology if and only if for some i, $1 \leq i \leq k$ and j, $1 \leq j \leq l$, $p_i = q_j$
- Given two clauses C_1 and C_2, $C_1 \models C_2$ if and only if all literals occurring in C_1 occur in C_2, that is C_1 subsumes C_2.
- Reduced CNF: no subsumption between clauses
- Subsumption can be eliminated in polynomial time (but it still may be too much work)
- Similar result for DNF
- CNF machine a device to convert φ to $\mathrm{cCNF}(\varphi)$
- CNF machine as a DNF machine

Complete Normal Forms

- A *minterm* is an elementary conjunction containing one literal from each set $\{p, \neg p\}$
- For every formula φ and every minterm t, either the formula $(\varphi \wedge t) \equiv t$ is a tautology, or $\varphi \wedge t$ is false.
- Given a valuation v of finite set of variables, t_v is conjunction of literals l such that $v(l) = 1$
- $d_\varphi = \bigvee_{v \in Mod(\varphi)} t_v$
- Each d_v entails φ
- For every formula φ, $\varphi \equiv d_\varphi$ is a tautology. Up to the order of variables, and the listing of minterms the representation $\varphi \mapsto d_\varphi$ is unique.

More on canonical forms

- Minterms: just rows in the table where the value is 1
- Maxclauses: clauses non-tautological clauses mentioning all variables
- Formula c_φ - conjunction of maxclauses entailed by φ
- For every formula φ, $\varphi \equiv c_\varphi$ is a tautology. Up to the order of variables, and the listing of maxclauses the representation $\varphi \mapsto c_\varphi$ is unique.

Consequences for Lindenbaum Algebra

- Equivalence classes of minterms are atoms in the Lindenbaum Algebra
- Since there are 2^n minterms, Lindedenbaum algebra has 2^n atoms (n - number of variables)
- Hence for finite set Var the Lindenbaum algebra of Var is a finite Boolean algebra
- Not every finite Boolean algebra is Lindenbaum algebra
- But if a finite Boolean algebra has 2^n atoms then it is a Lindenbaum algebra

Other normal forms

- Implication normal form (only implication, $\perp$ and variables)
- Hence by adding additional variables we can reduce every formula to collection of implications
- if-then-else normal form

Compactness of propositional logic

- König Lemma: an infinite, finitely splitting tree possesses an infinite branch
- (**Compactness Theorem**) If a set of formulas F is unsatisfiable then there is a *finite* subset $F_0 \subseteq F$ such that F is unsatisfiable
- Equivalently: when all finite subsets of F are satisfiable then F is satisfiable
- Constructing a tree associated with a family of clauses
- Corollary: The operator Cn is continuous

Resolution

- Clausal logic: logic where only formulas are clauses
- Alternative approach: negation and infinitely many functors, n-ary disjunctions, each with its own table
- Such logic expresses the same theories as the full propositional logic
- Since such disjunctions are commutative w.r.t. all permutations we treat clauses as sets of literals, no repetition
- Fundamental operation on clauses: resolution

$$\frac{l \vee C_1 \qquad\qquad\qquad\qquad \bar{l} \vee C_2}{C_1 \vee C_2}.$$

- Resolution rule is *sound*

Closure under Resolution

- Given a set of clauses F there is a least set of clauses G such that
  1. $F \subseteq G$
  2. Whenever D_1 and D_2 are two clauses in G and the operation $Res(D_1, D_2)$ is executable and results in a non-tautological clause then $Res(D_1, D_2)$ belongs to G.
- Operator res_F (in the complete lattice of subsets of the set of clauses)

$$
\begin{aligned}
res_F(G) = F \cup \{Res(C_1, C_2) : C_1, C_2 \in F \cup G \\
\text{and } Res(C_1, C_2) \text{ is defined and non-tautological}\}.
\end{aligned}
$$

- res_F has a least fixpoint. This is the closure under resolution

Derivations, proof-theoretic approach

- Derivation of a clause: a labeled binary tree: leaves labeled with elements of F, internal nodes labeled with resolvents
- Proof-theorists invert those trees. Root is at the bottom
- As every ordered tree, such trees can be represented as strings (with depth-first traversal)
- In this way we can define proofs as sequences of clauses:
  1. Conclusion of the rule application application occurs always later than premises
  2. If a clause occurs in the sequence then it belongs to F or is a conclusion of an application of resolution to earlier elements of the clause
  3. The proven clause is last element of the sequence
- Both approaches (trees, proofs) are equivalent
- Der_F set of clauses that have proof

Is resolution complete?

- Resolution alone is not complete. That is there are clauses semantically entailed, but not provable by resolution
- Reason: resolution does not add variables
- Example: $F := \{p \vee q\}$, $C := p \vee q \vee r$. Clearly $F \models C$, but the only thing we can prove (using resolution) is the only clause in F

Minimal Resolution Consequence

- Elements of $Res(F)$ - resolution consequences
- Inclusion-minimal elements of $Res(F)$ - minimal resolution consequences
- If C is a resolution consequence of F, then there must be minimal resolution consequence of F C' included in C

Subsumption rule

- Subsumption rule:

$$\frac{C}{C \vee D}$$

- Subsumption rule is sound
- We can define the notion of derivation (tree) using subsumption and resolution
- We can define the the notion of proof (sequence of clauses)
- Whatever can be proved with one can be proved with the other

Quine theorem, completeness

- A non-tautological clause C is a consequence of a set of clauses F if and only if there is a Resolution consequence of F, D, such that $D \subseteq C$.
- A non-tautological clause C is a consequence of a CNF F if and only if for some minimal Resolution consequence D of F, $D \subseteq C$.
- A CNF F is satisfiable if and only if $\emptyset \notin Res(F)$.
- (**Quine Theorem** Proof system consisting of Resolution and Subsumption rules is complete for clausal logic. That is, given a set of clauses F and a clause C, $F \models C$ if and only if there exists a derivation (using Resolution and Subsumption rules) that proves C
- We can limit the proof to admit a single application of Subsumption.

Resolution refutation

- Given a clause C,
$$\neg C = \{\bar{l} : l \in C\}$$
- If C is a clause then $\neg C$ is a set of unit clauses
- We say that C follows from F by *Resolution refutation* if the closure under Resolution of $F \cup \neg C$ contains an empty clause (i.e. is inconsistent).
- Let F be a CNF, and let C be a clause. Then $F \models C$ if and only if C follows from F by Resolution refutation.

The basis of resolution consequences

- Basis G of $Res(F)$
  1. $G \subseteq Res(F)$
  2. $Cn(F) = Cn(G)$
  3. G forms an antichain, i.e. for $C_1, C_2 \in G$, if $C_1 \subseteq C_2$ then $C_1 = C_2$
  4. Every element of $Res(F)$ is subsumed by some element of G
- Let F be a set of clauses. Then F possesses a unique basis
- How to compute a basis w/o computing the entire $Res(F)$?

Basis, cont'd

- Start with an CNF F. Due to the discussion above we can assume that F is subsumption-free.
- Non-deterministically, select from F a pair of resolvable clauses C_1 and C_2 which has not been previously resolved.
- If $Res(C_1, C_2)$ is subsumed by some other clause in F or is a tautology, do nothing and select next pair.
- If $D := Res(C_1, C_2)$ is *not* subsumed by some other clause in F, do two things:
  (a) Compute $R := \{E \in F : D \subseteq E\}$. Eliminate from F all clauses subsumed by D, that is $F := F \setminus R$
  (b) Set $F := F \cup \{D\}$.
- Do this until F does not change.

Basis, cont’d

- Two invariants:
  - Antichain
  - Consequences
- This implies termination

Another preprocessing for query ans.

- Pruning F, given C:

$$F/C = \{D : D \in F \text{ and } D \text{ can not be resolved with } C\}.$$

- $F \models C$ if and only if $F/C \models C$

Davis-Putnam reduct

- The reduct F_l is
$$F_l := \{C \setminus \{\bar{l}\} : C \in F \wedge l \notin C\}$$
- The set F splits into three sets: of clauses mentioning l, of clauses mentioning $\bar{l}$ and those not mentioning $|l|$ at all
- F_l arises from the second and the third
- Let F be a set of non-tautological clauses, and let l be a literal. Then F is unsatisfiable if and only if both F_l and $F_{\bar{l}}$ are unsatisfiable.
- Dual form: Let F be a set of non-tautological clauses, and let l be a literal. Then F is satisfiable if and only if at least one of F_l, $F_{\bar{l}}$ is satisfiable.

Free literal

- l is free for F if no clause of F contain $\bar{l}$
- Let F be a set of clauses. If l is a free literal in F then F is satisfiable if and only if F_l is satisfiable.
- Moreover, if v is a valuation satisfying F_l, then a valuation w defined by

$$w(m) = \begin{cases} v(m) & m \notin \{l, \bar{l}\} \\ 1 & m = l \end{cases}$$

satisfies F.

Semantic Resolution

- F given. Let us fix v. F splits in F_0 – clauses of F unsatisfied by v, F_1 clauses of F satisfied by v
- If F unsatisfiable then F_0, F_1 nonempty
- Ordering variables induces order of literals in each non-tautological clauses (via ordering of underlying variables)
- Semantic resolution rule (remember v fixed) applied to an *ordered* pair (D, E) of clauses of F
  (a) $v(D) = 1$
  (b) $v(E) = 0$
  (c) D, E are resolved on the literal largest in the ordering $\prec_E$ (i.e. highest in E)

Semantic Resolution, cont'd

- Operator $res_{F,v,\prec}(\cdot)$

$$res_{F,v,\prec}(G) = \{C : C \in F \vee \exists_{D,E}(D \in F \cup G \wedge \\ E \in F \cup G \wedge v(D) = 1 \wedge v(E) = 0 \wedge C = Res_{v,\prec}(D,E))\}.$$

- The operator $res_{F,v,\prec}$ is monotone. Thus it possesses a least fixpoint.
- $Res_{v,\prec}(F)$
- Completeness of semantic resolution; given F, v, $\prec$, F is unsatisfiable if and only if $Res_{v,\prec}(F)$ contains $\emptyset$

Testing satisfiability

- Table method
- Tableaux
- Clausal Logic: Davis-Putnam two-phase algorithm
- Clausal Logic: Davis-Putnam-Logemann-Loveland algorithm

Hintikka Sets

- No immediate contradiction ($\perp$, no contradictory atoms)
- Standard decomposition principles for $\neg$, $\wedge$, $\vee$
- Hintikka set is always consistent

Tableaux

- Binary trees with nodes decorated with signed formulas
- Tableau for a theory T
- Open and closed branches
- Managing branches via expansion rules
- Finished tableaux

Fundamental tableaux theorem

- Open branch in a finished tableau is a Hintikka set
- Thus if the a tableau for T has an open branch then T is satisfiable
- Canonical tableau for T
- T is satisfiable if and only if the canonical finished tableau for T has an open branch
- T is unsatisfiable if and only if the canonical finished tableau for T has no open branches
- (Completeness for tableaux). $T \models \varphi$ if and only if canonical finished tableau for $T \cup \{\neg\varphi\}$ is closed

Variable elimination

- Operation $F - l$, handling free literals
- Variable elimination resolution $Res_x(F)$
  - Select variable x
  - If $l = x$, or if $l = \bar{x}$ is free in F, $F - l$
  - O/w clauses without occurrence of x, and clauses obtained by resolving w.r.t x, tautologies eliminated
- Thus $Res_x(F)$ has no occurrence of x
- Clauses of F that have x maintained separately

Variable Elimination

- If v satisfies F then v satisfies $Res_x(F)$
- We will assume that we have some heuristic function $select(V)$ selecting a variable from an input set of variables V
- This heuristic funtion guides variable elimination resolution
- Decomposition of F w.r.t x:
  - $F^0 = \{C \in F : \neg x \in C\}$,
  - $F^1 = \{C \in F : x \in C\}$,
  - $F^2 = \{C \in F : x, \neg x \notin C\}$

Introducing DP algorithm, part 1

- Selection function assumed
- Three sequences: $\langle F_j \rangle$, $\langle x_j \rangle$, $\langle S_j \rangle$
  - $F_0 := F$, $x_0 := \mathit{select}_{\mathit{Var}_{F_0}}$, $S_0 = F_0^0 \cup F_0^1$, $F_1 = \mathit{Res}_{x_0}(F_0)$
  - If $\mathit{Var}_{F_j} = \emptyset$, halt.
  - O/w $x_j := \mathit{select}_{\mathit{Var}_{F_j}}$, $S_j = F_j^0 \cup F_j^1$,
    $F_{j+1} = \mathit{Res}_{x_j}(F_j)$
- Possible outcomes: $F_{n-1} = \emptyset$ or $F_{n-1} = \{\emptyset\}$
- In the latter case F is unsatisfiable because $\bigcup_i F_i$ is included in $\mathit{Res}(F)$
- We can stop at any moment once $\emptyset$ computed

Going back, part II of DP algorithm

- We assume part I did not return the string "input theory unsatisfiable"
- S_n is empty. Why?
- Two possible rasons:
    - The last variable selected occurred either positively in *all clauses* or negatively in *all clauses*
    - All we produced during the resolution w.r.t last variables were tautologies, so $Res_{v_{n-1}}(F_{n-1})$ is empty

Going back, part II of DP algorithm, cont'd

- The idea: construct a sequence of partial valuations, going *backwards*
- This partial valuation defined on all variables occurring in S_{n-1} and nothing else
- Base Case 1. x_{n-1} free in S_{n-1}. We set $v(x_{n-1} = 1$, all other variables still occurring in S_{n-1} zeroed (but we do not have to do so, b.t.w.)
- Base Case 2. $\neg x_{n-1}$ free in S_{n-1}. We set $v(x_{n-1} = 0$, all other variables still occurring in S_{n-1} zeroed (but we do not have to do so, b.t.w.)
- Base case 3. Some occurrences of x_{n-1} in S_{n-1} positive, some negative. Tricky!

Going back, part II of DP algorithm, cont'd

- Inductive case
- Reduct w.r.t. a partial valuation - eliminating clauses that are already satisfied, eliminating literals that can not be used for satisfaction
- Inductive assumption is that we satisfied all layers S_k, $k \geq j$ with a valuation v
- Goal: Satisfy S_{j-1}
- We reduce S_{j-1} by current v. Variables of current v no longer occur in the reduced CNF
- We define an extension of current v to new v so that v is defined on all variables of S_{j-1}, and those occurring in later S_k and nothing else. How?

Going back, part II of DP algorithm, cont'd

- Four, not three cases possible
  - Reduct is empty. That is all clauses in S_{j-1} already satisfied. We zero any variable occurring in S_{j-1} but not in current v
  - Literal x_{j-1} is free in the reduct. We set the value of x_{j-1} to 1, zero all other variable
  - Literal $\neg x_{j-1}$ is free in the reduct. We set the value of x_{j-1} to 0, zero all other variable
  - x_{j-1} occurs both positively and negatively in S_{j-1}. We proceed similarly to case 3 of base
- Proof of correctness of this procedure

Wrapping up DP

- The DP algorithm is complete. That is, if F is a set of clauses, and $select(\cdot)$ is a selection function, then:
  1. If F is satisfiable then the DP algorithm returns on the input F and the selection function $Select$ a valuation satisfying F
  2. If F is unsatisfiable, then the DP returns the string 'input formula unsatisfiable'.
- In the second phase we have plenty of leverage, we do not have to use anything already computed, but can plug in other partial valuations, as long as they satisfy the rest

The tree of partial valuations

- We assume a *finite* set of clauses
- We arrange *partial valuation* in a search tree (i.e. label full binary tree - but see below - with partial valuations)
- We assume that, like in DP algorithm we have a heuristic function $select(\cdot)$ that assigns to a partial valuation v a variable *outside* the domain of v (providing $Dom(v) \neq Var_F$)
- Then node n has two children: n_0, n_1. The first one labeled by $v \cup \{select(v)\}$, the other labeled by $v \cup \{\neg select(v)\}$

Pruning, BCP

- Given a set of literals v

$$bcp_F(S) = \{l : \text{There is } C := l_1 \vee \ldots \vee l_k \vee l \in F, \bar{l}_1, \ldots, \bar{l}_k \in v\}$$

- $bcp(\cdot)$ is a monotone operator in the complete lattice of sets of literals
- $\mathrm{BCP}(F)$ is the least fixpoint of the operator $bcp(\cdot)$
- Proof-theoretic representation of BCP, unit resolution.
- Both approaches equivalent

DPLL algorithm

- Let F be a CNF. Then F is satisfiable if and only if $\mathrm{BCP}(F)$ consistent and the reduct of F w.r.t. $\mathrm{BCP}(F)$ is satisfiable
- An obvious observation: Given a CNF F, F is satisfiable if and only if $F \cup \{p\}$ is satisfiable or $F \cup \{\neg p\}$ is satisfiable
- If G is the reduct of F by means of $\mathrm{BCP}(F)$ then $\mathrm{BCP}(G) = \emptyset$
- But often it may happen that $\mathrm{BCP}(F) = \emptyset$ but $\mathrm{BCP}(F \cup \{l\}) \neq \emptyset$
- DPLL: systematic backtracking search of the tree of partial valuations, with BCP as a pruning algorithm and user-provide selection function
- DPLL is complete
- Improvements to DPLL

Craig Lemma

- Given a valid implication $\psi \Rightarrow \varphi$, an *interpolant* is any formula ϑ such that both $\psi \Rightarrow \vartheta$ and $\vartheta \Rightarrow \varphi$ are valid
- Craig Lemma: Given a valid implication $\psi \Rightarrow \varphi$, there is an interpolant ϑ so that variables in ϑ occur *both* in φ and in ψ
- Stronger form will be shown
- Recent reports of the use of Craig Lemma in Bounded Model Checking

Few lemmas etc.

- If ψ is DNF, $\psi := D_1 \vee \ldots \vee D_k$ then $\psi \Rightarrow \varphi$ is a tautology if and only if for all i, $1 \leq i \leq k$, $D_i \Rightarrow \varphi$ is a tautology
- If D is an elementary conjunction and X set of variables, then $D \mid_X$ is the result of eliminating from D literals based on variables not in X
- Let D be a noncontradictory elementary conjunction, let φ be an arbitrary formula. Then $D \Rightarrow \varphi$ is a tautology if and only if $D \mid_{Var_\varphi} \Rightarrow \varphi$ is a tautology
- Thus, assuming $\psi \Rightarrow \varphi$ is a tautology, $\psi := D_1 \vee \ldots \vee D_k$ a DNF, all D_i non-contradictory. Then

  $$D_1 \mid_{Var_\varphi} \vee \ldots D_k \mid_{Var_\varphi} \Rightarrow \varphi$$

  is a tatutolgy

Craig Lemma

- For every formula φ, the formula

$$D_1 \vee \ldots \vee D_k \Rightarrow D_1 \mid_{\mathrm{Var}(\varphi)} \vee \ldots \vee D_k \mid_{\mathrm{Var}(\varphi)}$$

  is a tautology.
- If $\psi \Rightarrow \varphi$ is a tautology, then there exists a formula ϑ such that $\mathrm{Var}(\vartheta) \subseteq \mathrm{Var}(\psi) \cap \mathrm{Var}(\varphi)$ and such that both $\psi \Rightarrow \vartheta$ and $\vartheta \Rightarrow \varphi$ are tautologies

Improving interpolant

- If φ is a formula with only positive occurrences of variable p then there are two formulas: ψ_1 and ψ_2 with no occurrence of p so that

$$\varphi \Leftrightarrow ((p \wedge \psi_1) \vee \psi_2)$$

  is a tautology

- If φ is a formula with only negative occurrences of variable p then there are two formulas: ψ_1 and ψ_2 with no occurrence of p so that

$$\varphi \Leftrightarrow ((\neg p \wedge \psi_1) \vee \psi_2)$$

  is a tautology

Eliminating literals

- Assume that $l_1 \wedge \ldots \wedge l_k \Rightarrow \varphi$ is a tautology, and l_1 does not occur in φ. Then $l_2 \wedge \ldots \wedge l_k \Rightarrow \varphi$ is a tautology
- Let D be an noncontradictory elementary conjunction, and let us assume that $D \Rightarrow \varphi$ is a tautology. Let D' be the conjunction of those literals in D which occur in φ. Then $D' \Rightarrow \varphi$ is a tautology
- (**Craig Lemma, strong form**) If $\psi \Rightarrow \varphi$ is a tautology then there is an interpolant ϑ with the property that whenever a variable occurs in ϑ then it occurs in ψ, φ (and also in ϑ) with the same polarities.

Is DNF in predecessor needed?

- Of course not, if we have CNF of the consequent we can do the same
- We can either transform $\psi \Rightarrow \varphi$ into $\neg\varphi \Rightarrow \neg\psi$ and use the previos technique, or prove analogous lemmata for CNF in the consequent
- We can also compute a "canonical candidate for interpolant”

Satisfaction for positive and negative formulas

- Formula φ is *positive* if it has no negative occurrences of variables
- Formula φ is *negative* if it has no positive occurrences of variables
- Valuation $\mathbf{1}$ satisfies positive formulas and so sets of positive formulas are satisfiable.
- Valuation $\mathbf{0}$ satisfies negative formulas and so sets of negative formulas are satisfiable.

Satisfying Horn formulas

- A Horn clause: at most one positive literal
- Basic classification of Horn clauses
  - facts
  - program clauses
  - constraints

Horn theories and families closed under intersection

- A family $\mathcal{F}$ of sets is closed under intersections if for every nonempty $\mathcal{Y}$, $\mathcal{Y} \subseteq \mathcal{X}$, $\bigcap \mathcal{Y}$ belongs to $\mathcal{X}$
- Families of sets closed under intersections appear all over CS and Mathematics
- We will use representation of valuations as *sets of variables*
- Let At be a finite set of atoms and let $\mathcal{S} \subseteq \mathcal{P}(At)$ be a nonempty family of sets. Then $\mathcal{S}$ is of the form $Mod(H)$ for some collection H of Horn clauses over At if and only if $\mathcal{S}$ is closed under intersections.

Program clauses and constraints

- Program clause: Horn clause with exactly one positive literal
- Constraint clause: Horn clause with no positive literals
- CNFs consisting of program clauses are satisfiable
- CNFs consisting of program clauses has a least model
- CNFs consisting of constraint clauses are satisfiable

Model existence for Horn theories

- Each Horn theory H decomposes into program part H_1 and constraint part H_2
- If H is Horn, then H is satisfiable if and only if least model of H_1 satisfies H_2
- Associating an operator O_H with the Horn program H
- O_H is monotone and continuous (regardless of the size of Var)
- Least model of the Horn program H coincides with the least fixpoint of O_H

Representability of monotone operators

- If Var is finite then for every monotone operator O in $\mathcal{P}(Var)$ there is program H such that $O = O_H$
- Dowling-Gallier algorithm for computation of least model of H for programs H
- Testing satisfiability of Horn H in linear time
- Completeness of unit resolution for querying the least model of Horn theory

Dual Horn formulas

- Dual Horn clause – at most one negative literal
- Permutation Inv
- Permutation Inv is consistent and transforms Horn formulas to dual Horn and conversely
- $M \models F$ if and only if $Var \setminus M \models Inv(F)$
- Inv is complement operation
- Inv acts on families of sets, and transforrms families closed under intersections to families closed under unions

Dual Horn formulas

- Various results on Horn theories are lifted to dual-Horn case
- BUT: operator associated with dual-Horn theories is monotone
- There is largest fixpoint, and this is the largest model of dH
- Unit resolution complete for dual-Horn theories, but now one of inputs can be required to be negative literal c

Krom formulas, 2SAT

- 2-clause, or *Krom clause*, a clause with at most two literals
- The set of 2-clauses is closed under resolution
- There is $O(n^2)$ 2-clauses
- Thus in DP the space used by the algorithm (and thus time) is bounded by polynomial
- DP solves the satisfiability problem for sets of 2-clauses in polynomial time (but we will see better performance)

A closure property of sets of 2-clauses

Let K be a 2-CNF, and let l be a literal such that $l, \bar{l} \notin \mathrm{BCP}(K)$. Let $v = \mathrm{BCP}(K \cup \{l\})$, and let $K_1 = \mathrm{reduct}(K, v)$. Then:

1. $K_1 \subseteq K$
2. If v is consistent then: K is consistent if and only if K_1 is consistent
3. If v is consistent then a satisfying valuation for K can be computed from v and a satisfying valuation for K_1 in linear time.

BCP and 2-SAT

- A partial valuation v *touches* clause C if $Var_C \cap Var_v \neq \emptyset$
- If F consists of 2-clauses then for every literal l, if $w = \mathrm{BCP}(F \cup \{l\})$ is consistent then w satisfies every clause in F which w touches
- If F consists of 2-clauses then F is satisfiable if and only if for every literal l, at least one of $\mathrm{BCP}(F \cup \{l\})$, $\mathrm{BCP}(F \cup \{\bar{l}\})$ is satisfiable
- Autarkies for Krom formulas

The graph G_F

- After initialization the resulting formula F is a subset of F and each clause has exactly 2 literals
- Vertices of G_F: Literals of Var_F
- Edges of G_F: whenever $l \vee m$ belongs to F, generate two edges: $(\bar{l}, m)$, $(\bar{m}, l)$
- G_F is a directed graph

Strong components and 2-CNF

- A strong component S of a directed graph: for every $x, y \in S$, $x \neq y$, there is a directed cycle containing both x and y
- If l, m are in same strongly connected component of G_F then

$$\mathrm{BCP}(F \cup \{l\}) = \mathrm{BCP}(F \cup \{m\})$$

- F a 2-CNF, all clauses of length 2. Then F is satisfiable if and only if no strong connected component of G_F contains a pair of dual literals
- Complexity of testing satisfiability of 2-CNF

Renameable variants of classes of formulas

- Given $\mathcal{C}$, a class of CNFs, renameable $\mathcal{C}$, all formulas which can be obtained from formulas of C by consistent permutations of literals
- Shift permutation: does not change underlying variable, may only change sign
- F is renameable Horn if for some renaming π, $\pi(F)$ is Horn, thus if for some shift π, $\pi(F)$ is Horn

Describing shift permutations

- There is a one-to-one correspondence between shift permutations and sets of atoms of the form $\mathit{shift}(x)$
- Representing shifting into Horn clauses Given a clause

$$C := p_1 \vee \ldots \vee p_k \vee \neg q_1 \vee \ldots \vee \neg q_l$$

Define a collection S_C of 2-clauses consisting of three groups:

Group 1. $\mathit{shift}(p_i) \vee \mathit{shift}(p_j), 1 \leq i < j \leq k$

Group 2. $\neg \mathit{shift}(q_i) \vee \neg \mathit{shift}(q_j), 1 \leq i < j \leq l$

Group 3. $\neg \mathit{shift}(q_i) \vee \mathit{shift}(p_j), 1 \leq i \leq l, 1 \leq j \leq k.$

Carrying on shifting

- $S_F = \bigcup_{C \in F} S_C$
- S_F is 2-CNF
- There is a one-to-one correspondence between valuations satisfying S_F and shifts of F into a Horn CNF
- Testing if F is renameable Horn is polynomial, in fact quadratic in F
- F is renameable Horn if and only if it is renameable dual Horn

Linear formulas

- Operation $\oplus$, a.k.a. XOR
- The structure $\langle \mathcal{BOOL}, \oplus, \wedge, \bot, \top \rangle$ is a field, called $\mathbb{Z}_2$
- Linear equation over that field: formula in $x_1, \ldots, x_n$ and possibly $\top$
- Special form of Gauss-Jordan elimination over $\mathbb{Z}_2$ – add it!
- Issues with the number of equations
- Triangular form of a set of equations (first pass, Gauss)
- Substitute (second pass, Jordan)

All you need is 3-CNF

- 3-clauses
- With additional variables (linear number of variables) we can reduce to 3-clauses. Specifically: for every finite set of formulas F there is a set of 3-clauses G such that $Var_G \supseteq Var_F$, Var_G contains a special variable n_F and there is a one-to-one correspondence between satisfying valuations for F and satisfying valuations for G that evaluate n_F as 1

Combinatorial circuits

- Treating a set of formulas as an input-output devise
- Acyclic digraphs with at most two parents per node
- Inputs and outputs
- Circuit representing a collection of formulas (one output per formula)
- 3-CNF representation of a circuit

Some complexity issues

- The problem SAT (i.e. the language consisting of satisfiable formuals) is NP-complete (proof in the notes, via coding of accepting computations of Turing machines where the length of operation is bounded by a fixed polynomial in the length of the input)
- Thus, the problem UNSAT is co-NP-complete
- By our reduction, the problem 3-SAT (satisfiability of the sets of clauses consisting of 3-clauses) is also NP-complete
- Thus, the problem 3-UNSAT is also co-NP-complete
- All *nontrivial* combinations of “easy” classes considered in this lecture also result in NP-complete classes (for instance: unions of Horn and dual-Horn are NP-complete)
- But there is *plenty* of other “easy classes”