Equational Reasoning

Pete Manolios
Northeastern

Logic and Computation, 1/31/2019

Objectives

- Review Equational Reasoning
- Decision Procedures
- Complete Boolean Bases
- DNF/CNF

Equational Proofs

- Which is the simplest expression equivalent to $\neg(p \oplus p \Rightarrow q)$?

```
  ¬(p ⊕ (p ⇒ q))
≡  { ¬(p⊕q) ≡ (p≡q)}
   (p ≡ (p ⇒ q))
≡  { Shannon }
   (p ∧ (true ≡ (true ⇒ q))) ∨ (¬p ∧ (false ≡ (false ⇒ q)))
≡  { Constant Prop }
   (p ∧ q) ∨ (¬p ∧ false)
≡  { Constant Prop }
   p ∧ q
```

Equational Proofs

- We are going to use equational proofs throughout the semester!
- An equational proof is just a sequence of equality preserving transformations
- To show that `f=g` is valid, we have a proof of the form:

```
   f
=  { hint 1 }
   f₁
=  { hint 2 }
   . . .
=  { hint n+1 }
   g
```

- If `g` is a validity (e.g., `true`), then this is a proof that `f` is valid
- Hints should contain enough information to understand the equality

Equational Proofs

- If the formulas are Boolean, we can instead use this form

```
   f
≡  { hint 1 }
   f₁
≡  { hint 2 }
   . . .
≡  { hint n+1 }
   g
```

- By transitivity of = and ≡
  - if f = (≡) f_1 and … and f_n = (≡) g, then
  - then f = (≡) g

Equational Proofs

- If we have a transitive operator, say ⇒, then this also works

```
   f
⇒  { hint 1 }
   f1
⇒  { hint 2 }
   . . .
⇒  { hint n+1 }
   g
```

- Can mix in ≡'s. By transitivity of ⇒ and ≡
  - if f ⇒/≡ f_1 and … and f_n ⇒/≡ g, then $f \Rightarrow g$
- Other transitive operators include <, >, ≤, ≥, etc.

ACL2s Decision Procedure

- ACL2s is a decision procedure for propositional validities
- Consider: $a \wedge b \equiv a \equiv b \equiv a \vee b$
- You can use ACL2s to check if this is valid:

```
(thm (implies (and (booleanp a)
                   (booleanp b))
              (iff (iff (and a b) a)
                   (iff b (or a b)))))
```

- Notice hypotheses are needed because?
- They're not needed due to contract completion (iff, and, or can be applied to All)
- The ACL2s universe contains more than Booleans and we want to make a claim about Booleans
- But, in ACL2s the above is a theorem even without the hypotheses

Complete Boolean Base

- Consider *f*, an arbitrary Boolean function of arity *n*
- How many functions of arity *n* are there?
  - the truth table for any such function has 2^n rows
  - each row has 2 possible values, so $2^{(2^n)}$ such functions
  - e.g, if *n*=5, then 2^{32} = 4,294,967,296 such functions
- Can we represent *all* Boolean functions with the operators we have?
- Yes. Take the disjunction of all the assignments that make *f* true
  - these assignments are just the rows in the truth table for which *f* is T
  - such assignment are *conjunctive clauses:* a conjunction of *literals*, atoms or their negations
  - to represent *f*, we take the disjunction of all the conjunctive clauses

Complete Boolean Base

- Consider $p \oplus q$. Here is the truth table:
- The $\vee$ of all the assignments that make *f* T:
  - $(p \wedge \neg q) \vee (\neg p \wedge q)$
- Notice we only used $\vee$, $\wedge$ and $\neg$
- $\{\vee, \wedge, \neg\}$ is a complete Boolean base
- If a set of Boolean operators can represent any Boolean function, it is a *complete Boolean base*
- Is there a simpler complete Boolean base?
- Yes! Both $\{\vee, \neg\}$ and $\{\wedge, \neg\}$ are complete
- Because $p \wedge q \equiv \neg(\neg p \vee \neg q)$ and $p \vee q \equiv \neg(\neg p \wedge \neg q)$ (DeMorgan)

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

DNF & CNF

- Disjunctive Normal Form (DNF): a disjunction of conjunctive clauses
  - e.g., true, p, q, $p \vee q$, $p \wedge q$, $(p \wedge \neg q) \vee (\neg p \wedge q)$
  - notice: at most a 2-level formula over *literals* (atoms or their negations)
- Conjunctive Normal Form (CNF): a conjunction of *clauses*, a disjunction of literals
  - e.g., true, p, q, $p \vee q$, $p \wedge q$, $(\neg p \vee \neg q) \wedge (p \vee q)$
- Given any function, we obtain CNF by taking the conjunction of the negation of assignments that make *f* false
  - e.g., consider $\oplus$
  - we get the negation of $(p \wedge q) \equiv \neg p \vee \neg q$
  - and the negation of $(\neg p \wedge \neg q) \equiv p \vee q$
  - to wind up with $(\neg p \vee \neg q) \wedge (p \vee q)$
  - the DNF was $(p \wedge \neg q) \vee (\neg p \wedge q)$

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

The Size of DNF/CNF

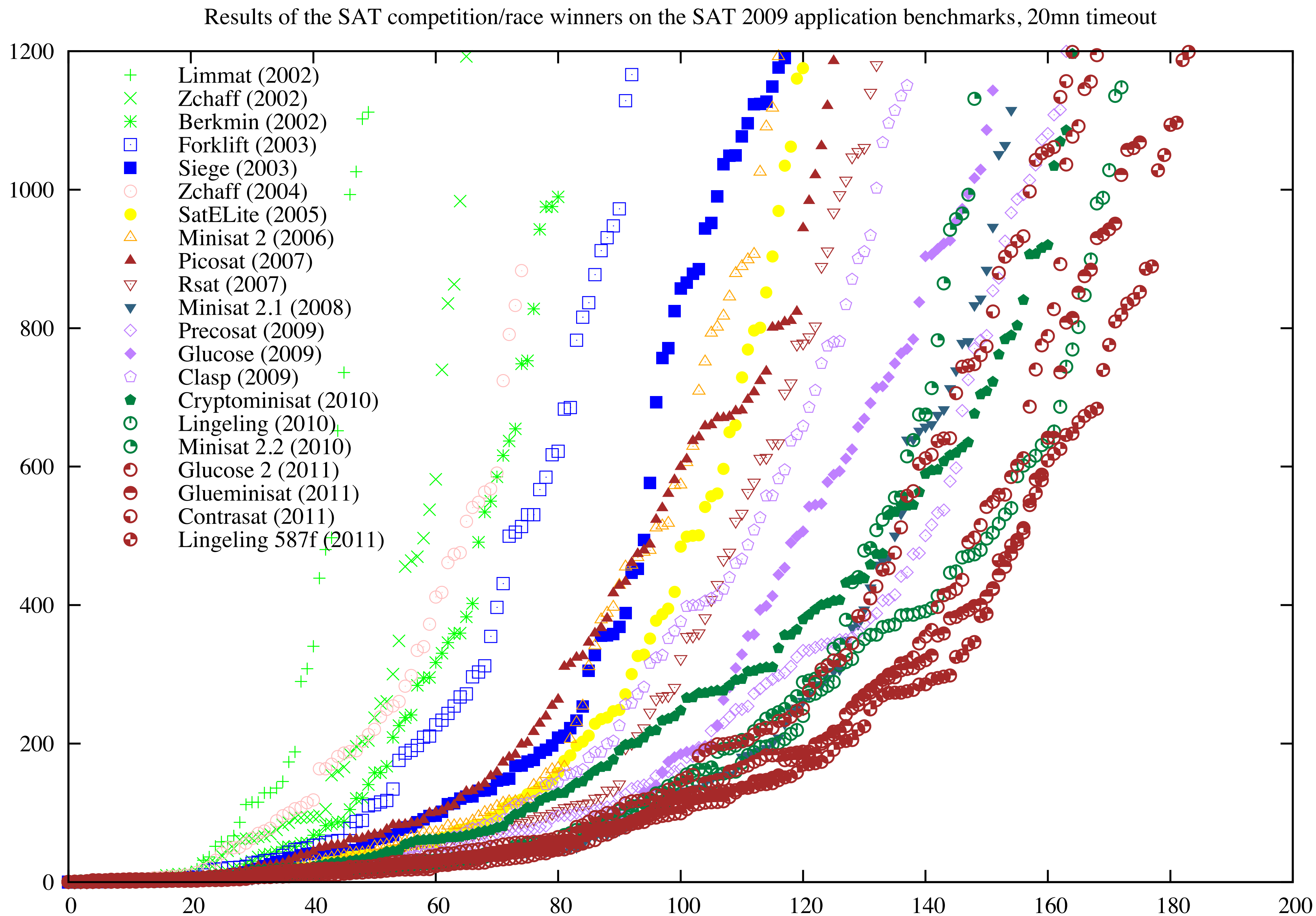
- There can be many equivalent DNFs
- Consider the function f
- Our DNF construction gives us
  - a disjunction of 6 conjunctive clauses, each involving p,q,r
- Is there a simpler DNF?
  - yes: $\neg p \vee q$
- So, DNF can be exponentially smaller than a truth table
  - great!
- Quiz: consider the formula $(a \vee b) \wedge (c \vee d) \wedge (e \vee f) \wedge (g \vee h)$ (has 4 clauses)
- The minimal DNF for this formula has how many conjunctive clauses?
  - **A: 1** **B: 3**
  - **C: 6** **D: 8**
  - **E: 16** **F: 64**

p	q	r	f
T	T	T	T
T	T	F	T
T	F	T	F
T	F	F	F
F	T	T	T
F	T	F	T
F	F	T	T
F	F	F	T

Normal Forms

- Minimizing DNF has many applications
  - this is used to analyze the reliability of safety-critical systems
- CNF is the input format of modern SAT solvers
  - this is the so-called DIMACS format
  - modern SAT solvers can solve industrial problems with 1M variables
- There are many other “normal” forms for Boolean formulae
  - decision trees: widely used in machine learning
  - BDDs: very powerful representation used in verification, AI, program analysis, …

SAT Cactus Plots



From: Le Berre&Biere 2011

Slides by Pete Manolios for CS2800, Logic & Computation, NU 2019