Prime Compilation of Non-Clausal Formulae

Joao Marques-Silva
Joint work with A. Previti, A. Ignatiev and A. Morgado

To be presented at IJCAI 2015

INESC-ID, IST, ULisbon, Portugal
CASL, CSI, UCD, Dublin, Ireland

Symposium on New Frontiers in Knowledge Compilation

VCLA, Vienna, Austria, June 2015

The success of SAT

- Well-known NP-complete decision problem [C71]

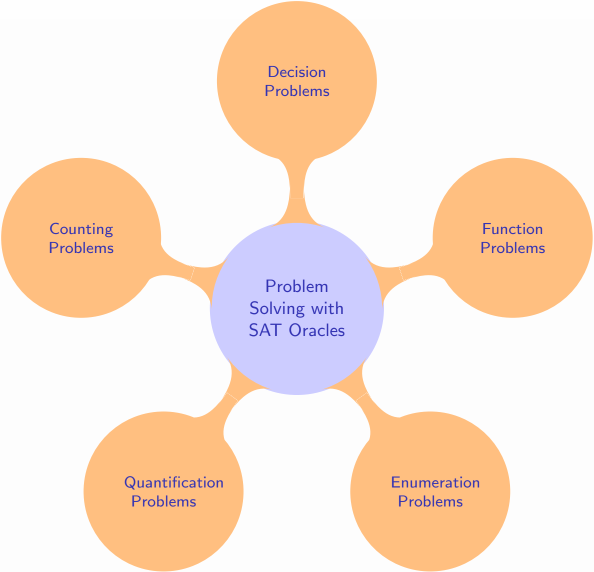
The success of SAT

- Well-known NP-complete decision problem [C71]
- In practice, SAT is a success story of Computer Science
  - Hundreds (even more?) of practical applications

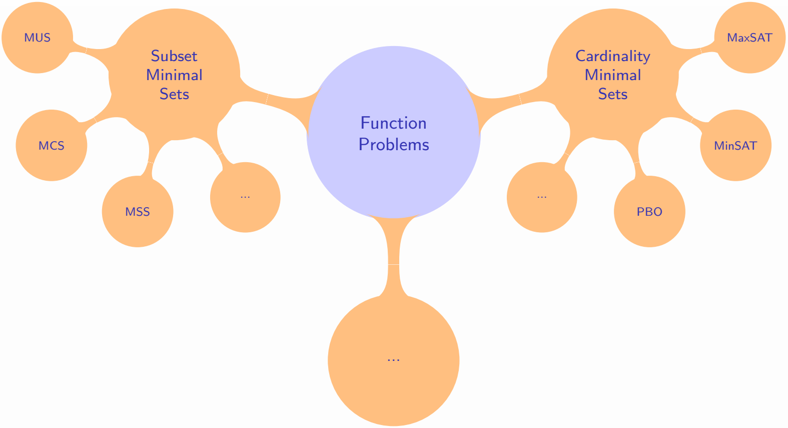
The success of SAT

- Well-known NP-complete decision problem [C71]
- In practice, SAT is a success story of Computer Science
  - Hundreds (even more?) of practical applications

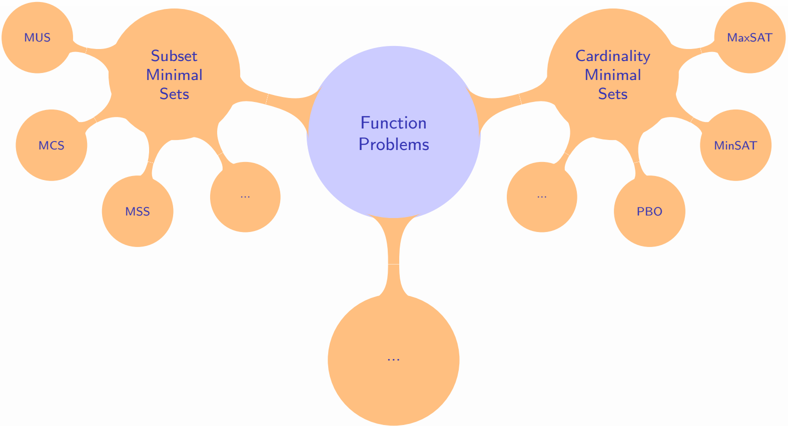
Problem solving with SAT oracles



Function problems



Function problems



- But also backbones, autarkies, MES, primes, etc.

An example – MUSes

$$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad (x_5 \vee x_6) \quad (\bar{x}_3 \vee \bar{x}_4) \quad (x_2) \quad (x_3) \quad (x_4)$$

- Formula is unsatisfiable but **not** irreducible

An example – MUSes

$$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad \boxed{(x_5 \vee x_6)} \quad (\bar{x}_3 \vee \bar{x}_4) \quad (x_2) \quad (x_3) \quad (x_4)$$

- Formula is unsatisfiable but **not** irreducible
- Can remove clauses, and formula still unsatisfiable

An example – MUSes

$\boxed{(\bar{x}_1 \vee \bar{x}_2)}\ \boxed{(x_1)}\ (x_5 \vee x_6)\ \ (\bar{x}_3 \vee \bar{x}_4)\ \boxed{(x_2)}\ (x_3)\ \ (x_4)$

- Formula is unsatisfiable but **not** irreducible
- Can remove clauses, and formula still unsatisfiable

- **Minimal Unsatisfiable Subset** (**MUS**):
  - Irreducible subformula that is unsatisfiable
    - MUSes are minimal sets

An example – MUSes

$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad (x_5 \vee x_6) \quad \boxed{(\bar{x}_3 \vee \bar{x}_4)} \quad (x_2) \quad \boxed{(x_3)}\,\boxed{(x_4)}$

- Formula is unsatisfiable but **not** irreducible
- Can remove clauses, and formula still unsatisfiable

- **Minimal Unsatisfiable Subset** (**MUS**):
  - Irreducible subformula that is unsatisfiable
    - MUSes are minimal sets

An example – MUSes

$(\bar{x}_1 \vee \bar{x}_2)$ (x_1) $(x_5 \vee x_6)$ $\boxed{(\bar{x}_3 \vee \bar{x}_4)}$ (x_2) $\boxed{(x_3)}$ $\boxed{(x_4)}$

- Formula is unsatisfiable but **not** irreducible
- Can remove clauses, and formula still unsatisfiable

- **Minimal Unsatisfiable Subset** (**MUS**):
  - Irreducible subformula that is unsatisfiable
    - MUSes are minimal sets

- Many applications: abstraction in software verification; debugging declarative models; pinpointing in DLs; type error debugging; etc.

An example – MCSes

$$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad (x_5 \vee x_6) \quad (\bar{x}_3 \vee \bar{x}_4) \quad (x_2) \quad (x_3) \quad (x_4)$$

- Formula is unsatisfiable with satisfiable subformulas

An example – MCSes

$\boxed{(\bar{x}_1 \vee \bar{x}_2)}$ (x_1) $\boxed{(x_5 \vee x_6)}$ $\boxed{(\bar{x}_3 \vee \bar{x}_4)}$ (x_2) (x_3) $\boxed{(x_4)}$

- Formula is unsatisfiable with satisfiable subformulas
- Can remove clauses such that remaining clauses are satisfiable

An example – MCSes

$\boxed{(\bar{x}_1 \vee \bar{x}_2)}$ (x_1) $(x_5 \vee x_6)$ $(\bar{x}_3 \vee \bar{x}_4)$ (x_2) (x_3) $\boxed{(x_4)}$

- Formula is unsatisfiable with satisfiable subformulas
- Can remove clauses such that remaining clauses are satisfiable

- **Minimal Correction Subset** (**MCS**):
  - Irreducible subformula such that the complement is satisfiable
    - MCSes are minimal sets

An example – MCSes

$$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad (x_5 \vee x_6) \quad (\bar{x}_3 \vee \bar{x}_4) \quad \boxed{(x_2)}\boxed{(x_3)} \quad (x_4)$$

- Formula is unsatisfiable with satisfiable subformulas
- Can remove clauses such that remaining clauses are satisfiable
- **Minimal Correction Subset** (**MCS**):
  - Irreducible subformula such that the complement is satisfiable
    - MCSes are minimal sets

An example – MCSes

$$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad (x_5 \vee x_6) \quad (\bar{x}_3 \vee \bar{x}_4) \quad \boxed{(x_2)} \quad (x_3) \quad \boxed{(x_4)}$$

- Formula is unsatisfiable with satisfiable subformulas
- Can remove clauses such that remaining clauses are satisfiable

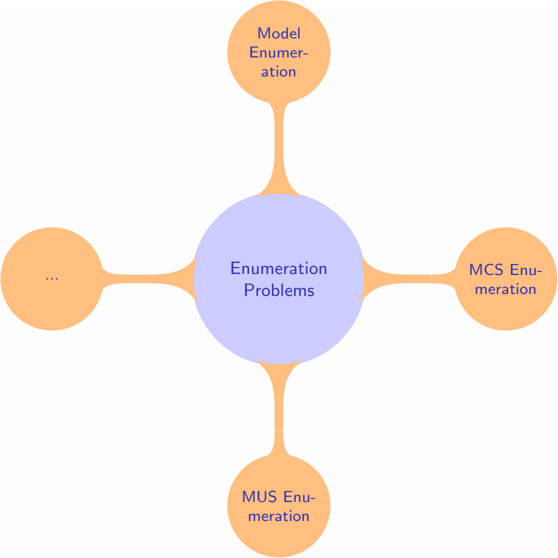
- **Minimal Correction Subset** (**MCS**):
  - Irreducible subformula such that the complement is satisfiable
    - MCSes are minimal sets

An example – MCSes

$$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad (x_5 \vee x_6) \quad (\bar{x}_3 \vee \bar{x}_4) \quad \boxed{(x_2)} \quad (x_3) \quad \boxed{(x_4)}$$

- Formula is unsatisfiable with satisfiable subformulas
- Can remove clauses such that remaining clauses are satisfiable
- **Minimal Correction Subset** (**MCS**):
  - Irreducible subformula such that the complement is satisfiable
    - MCSes are minimal sets
- Many applications: restore consistency; smallest MCSes are MaxSAT solutions; MUS enumeration; minimal/maximal models; etc.

Enumeration problems



An example – MCS&MUS enumeration

- MCS enumeration is easy:
  - Extract & block MCSes, e.g. with MaxSAT or dedicated algorithm

An example – MCS&MUS enumeration

- MCS enumeration is easy:
  - Extract & block MCSes, e.g. with MaxSAT or dedicated algorithm
- MUS enumeration is (apparently) hard:
  - Unclear how to block MUSes

An example – MCS&MUS enumeration

- MCS enumeration is easy:
  - Extract & block MCSes, e.g. with MaxSAT or dedicated algorithm
- MUS enumeration is (apparently) hard:
  - Unclear how to block MUSes
  - Minimal hitting set dualization

An example – MCS&MUS enumeration

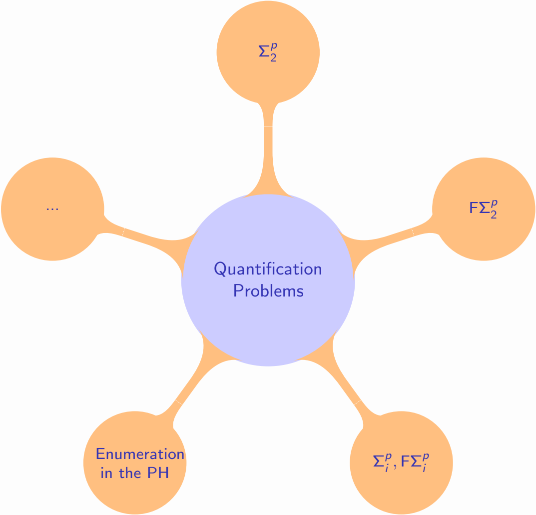
- MCS enumeration is easy:
  - Extract & block MCSes, e.g. with MaxSAT or dedicated algorithm
- MUS enumeration is (apparently) hard:
  - Unclear how to block MUSes
  - Minimal hitting set dualization
    - Explicit: find all MCSes and dualize

An example – MCS&MUS enumeration

- MCS enumeration is easy:
  - Extract & block MCSes, e.g. with MaxSAT or dedicated algorithm

- MUS enumeration is (apparently) hard:
  - Unclear how to block MUSes
  - Minimal hitting set dualization
    - Explicit: find all MCSes and dualize
    - Implicit: exploit hitting set dualization and iteratively find MCses and MUSes

Quantification



Application of enumeration – prime compilation

- Enumerate all prime implicates for:

$$(c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)$$

Application of enumeration – prime compilation

- Enumerate all prime implicates for:

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

  - Primes: $(c); (a \vee b)$

Application of enumeration – prime compilation

- Enumerate all prime implicates for:

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

  – Primes: $(c); (a \vee b)$

- Enumerate all prime implicants for:

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

Application of enumeration – prime compilation

- Enumerate all prime implicates for:

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

  - Primes: $(c); (a \vee b)$

- Enumerate all prime implicants for:

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

  - Primes: $(b \wedge c); (a \wedge c)$

Application of enumeration – prime compilation

- Enumerate all prime implicates for:

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

  - Primes: $(c); (a \vee b)$

- Enumerate all prime implicants for:

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

  - Primes: $(b \wedge c); (a \wedge c)$

- Enumerate all prime implicants for:

$$(((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

Application of enumeration – prime compilation

- Enumerate all prime implicates for:

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

  - Primes: $(c); (a \vee b)$

- Enumerate all prime implicants for:

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

  - Primes: $(b \wedge c); (a \wedge c)$

- Enumerate all prime implicants for:

$$(((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

  - Primes: $(b \wedge c); (a \wedge c)$

Application of enumeration – prime compilation

- Enumerate all prime implicates for:

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

  - Primes: $(c); (a \vee b)$

- Enumerate all prime implicants for:

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

  - Primes: $(b \wedge c); (a \wedge c)$

- Enumerate all prime implicants for:

$$(((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

  - Primes: $(b \wedge c); (a \wedge c)$

- Enumeration of primes studied since the **1930s!**
  - Formula minimization; Knowledge compilation; ...

Application of enumeration – prime compilation

- Enumerate all prime implicates for:

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

  - Primes: $(c); (a \vee b)$
- Enumerate all prime implicants for:

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

  - Primes: $(b \wedge c); (a \wedge c)$
- Enumerate all prime implicants for:

$$(((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

  - Primes: $(b \wedge c); (a \wedge c)$
- Enumeration of primes studied since the **1930s!**
  - Formula minimization; Knowledge compilation; ...
- **How to enumerate primes of non-clausal formulae, with SAT oracles?**

Outline

Outline

Propositional formulae

- Clausal:

Propositional formulae

- Clausal:
  - CNF: conjunction of disjunctions of literals

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

Propositional formulae

- Clausal:
  - CNF: conjunction of disjunctions of literals

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

  - DNF: disjunction of conjunctions of literals

$$(c \wedge a) \vee (c \wedge \neg a) \vee (a \wedge b \wedge d) \vee (a \wedge b \wedge \neg d)$$

Propositional formulae

- Clausal:
  - CNF: conjunction of disjunctions of literals

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

  - DNF: disjunction of conjunctions of literals

$$(c \wedge a) \vee (c \wedge \neg a) \vee (a \wedge b \wedge d) \vee (a \wedge b \wedge \neg d)$$

  - Other notation: Product of Sums (POS) / Sum of Products (SOP)

Propositional formulae

- Clausal:
  - CNF: conjunction of disjunctions of literals

$$(c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

  - DNF: disjunction of conjunctions of literals

$$(c \wedge a) \vee (c \wedge \neg a) \vee (a \wedge b \wedge d) \vee (a \wedge b \wedge \neg d)$$

  - Other notation: Product of Sums (POS) / Sum of Products (SOP)
- Non-clausal:
  - Non-CNF and non-DNF
  - Propositional formulae: well-formed formulae built with standard connectives $\neg$, $\wedge$, $\vee$

$$(((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

Defining primes

- Given formula F, a prime implicate is a non-empty set of non-complementary literals q, s.t.

$$F \models (\vee_{l \in q} l) \ \wedge \ \forall_{q' \subsetneq q} F \not\models (\vee_{l \in q'} l)$$

- Prime implicate q given implicate c, $q \subseteq c$

Defining primes

- Given formula F, a prime implicate is a non-empty set of non-complementary literals q, s.t.

$$F \vDash (\vee_{l \in q} l) \ \wedge \ \forall_{q' \subsetneq q} F \nvDash (\vee_{l \in q'} l)$$

- Prime implicate q given implicate c, $q \subseteq c$
- Given formula F, a prime implicant is a non-empty set of non-complementary literals p, s.t.

$$(\wedge_{l \in p} l) \vDash F \ \wedge \ \forall_{p' \subsetneq p} (\wedge_{l \in p'} l) \nvDash F$$

- Prime implicant p given implicant t, $p \subseteq t$

Defining primes

- Given formula F, a prime implicate is a non-empty set of non-complementary literals q, s.t.

$$F \vDash (\vee_{l \in q} l) \ \wedge \ \forall_{q' \subsetneq q} F \nvDash (\vee_{l \in q'} l)$$

- Prime implicate q given implicate c, $q \subseteq c$
- Given formula F, a prime implicant is a non-empty set of non-complementary literals p, s.t.

$$(\wedge_{l \in p} l) \vDash F \ \wedge \ \forall_{p' \subsetneq p} (\wedge_{l \in p'} l) \nvDash F$$

- Prime implicant p given implicant t, $p \subseteq t$
- **Each prime implicant (resp. implicate) of F is a minimal hitting set of the prime implicates (resp. implicants) of F** [R94]

Computing primes

- Extract one prime implicant for F in CNF:

Computing primes

- Extract one prime implicant for F in CNF:
  - Find satisfying assignment μ of F

Computing primes

- Extract one prime implicant for F in CNF:
  - Find satisfying assignment μ of F
  - Drop literals from μ while F satisfied

Computing primes

- Extract one prime implicant for F in CNF:
  - Find satisfying assignment μ of F
  - Drop literals from μ while F satisfied

- Similar for prime implicate with F in DNF and falsifying assignment

Computing primes

- Extract one prime implicant for F in CNF:
  - Find satisfying assignment μ of F
  - Drop literals from μ while F satisfied
- Similar for prime implicate with F in DNF and falsifying assignment
- **How about the general case of prime implicates for CNF, prime implicants for DNF, or primes for non-clausal?**
- **And, how about enumeration of primes?**
  - Repeated application of procedure above does not work...

Defining MUSes/MCSes/MSSes

- Given CNF F, with $F \vDash \perp$:

Defining MUSes/MCSes/MSSes

- Given CNF F, with $F \vDash \bot$:
  - $M \subseteq F$ is a Minimal Unsatisfiable Subset (MUS) iff:

$$M \vDash \bot \ \wedge \ \forall_{M' \subsetneq M} M' \nvDash \bot$$

Defining MUSes/MCSes/MSSes

- Given CNF F, with $F \vDash \bot$:
  - $M \subseteq F$ is a Minimal Unsatisfiable Subset (MUS) iff:

$$M \vDash \bot \ \land \ \forall_{M' \subsetneq M} \, M' \nvDash \bot$$

  - $S \subseteq F$ is a Maximal Satisfiable Subset (MSS) iff:

$$S \nvDash \bot \ \land \ \forall_{S \subsetneq S' \subseteq F} \, S' \vDash \bot$$

Defining MUSes/MCSes/MSSes

- Given CNF F, with $F \vDash \perp$:
  - $M \subseteq F$ is a Minimal Unsatisfiable Subset (MUS) iff:

    $$M \vDash \perp \ \wedge \ \forall_{M' \subsetneq M} M' \nvDash \perp$$

  - $S \subseteq F$ is a Maximal Satisfiable Subset (MSS) iff:

    $$S \nvDash \perp \ \wedge \ \forall_{S \subsetneq S' \subseteq F} S' \vDash \perp$$

  - $C \subseteq F$ is a Minimal Correction Subset (MCS) iff:

    $$F \setminus C \nvDash \perp \ \wedge \ \forall_{C' \subsetneq C} F \setminus C' \vDash \perp$$

Defining MUSes/MCSes/MSSes

- Given CNF F, with $F \vDash \perp$:
  - $M \subseteq F$ is a Minimal Unsatisfiable Subset (MUS) iff:

    $$M \vDash \perp \wedge \forall_{M' \subsetneq M} M' \nvDash \perp$$

  - $S \subseteq F$ is a Maximal Satisfiable Subset (MSS) iff:

    $$S \nvDash \perp \wedge \forall_{S \subsetneq S' \subseteq F} S' \vDash \perp$$

  - $C \subseteq F$ is a Minimal Correction Subset (MCS) iff:

    $$F \setminus C \nvDash \perp \wedge \forall_{C' \subsetneq C} F \setminus C' \vDash \perp$$

  - An MCS C is the complement (wrt to F) of an MSS S, $C = F \setminus S$

Defining MUSes/MCSes/MSSes

- Given CNF F, with $F \vDash \perp$:
  - $M \subseteq F$ is a Minimal Unsatisfiable Subset (MUS) iff:
    $$M \vDash \perp \wedge \forall_{M' \subsetneq M} M' \nvDash \perp$$
  - $S \subseteq F$ is a Maximal Satisfiable Subset (MSS) iff:
    $$S \nvDash \perp \wedge \forall_{S \subsetneq S' \subseteq F} S' \vDash \perp$$
  - $C \subseteq F$ is a Minimal Correction Subset (MCS) iff:
    $$F \setminus C \nvDash \perp \wedge \forall_{C' \subsetneq C} F \setminus C' \vDash \perp$$
  - An MCS C is the complement (wrt to F) of an MSS S, $C = F \setminus S$
  - **Each MCS (resp. MUS) of F is a minimal hitting set of the MUSes (resp. MCSes) of F** [R'87,BL'03,BS'05,LS'08]

Working with groups – MUSes

- Group of clauses 0, G_0, denoting a set of background (or don't care) clauses

Working with groups – MUSes

- Group of clauses 0, G_0, denoting a set of background (or don't care) clauses
- Group of clauses i, G_i
- Set of groups of clauses $\Gamma = \{G_1, \ldots, G_k\}$

Working with groups – MUSes

- Group of clauses 0, G_0, denoting a set of background (or don't care) clauses
- Group of clauses i, G_i
- Set of groups of clauses $\Gamma = \{G_1, \ldots, G_k\}$
- Conjunction of clauses in all groups unsatisfiable:

$$\bigwedge_{\substack{G_i \in G_0 \cup \Gamma \\ c \in G_i}} (c) \vDash \bot$$

Working with groups – MUSes

- Group of clauses 0, G_0, denoting a set of background (or don't care) clauses
- Group of clauses i, G_i
- Set of groups of clauses $\Gamma = \{G_1, \ldots, G_k\}$
- Conjunction of clauses in all groups unsatisfiable:

$$\bigwedge_{\substack{G_i \in G_0 \cup \Gamma \\ c \in G_i}} (c) \vDash \perp$$

- Group MUS, $\Psi \subseteq \Gamma$:

$$\bigwedge_{\substack{G_i \in G_0 \cup \Psi \\ c \in G_i}} (c) \vDash \perp \;\wedge\; \forall_{\Psi' \subsetneq \Psi} \bigwedge_{\substack{G_i \in G_0 \cup \Psi' \\ c \in G_i}} (c) \nvDash \perp$$

Reducing primes to group MUSes – prime implicates

- Recall definition of prime implicate $p \subseteq c$:

$$F \models (\vee_{l \in q} l) \;\wedge\; \forall_{q' \subsetneq q} F \not\models (\vee_{l \in q'} l)$$

Reducing primes to group MUSes – prime implicates

- Recall definition of prime implicate $p \subseteq c$:

$$F \models (\vee_{l \in q} l) \;\wedge\; \forall_{q' \subsetneq q} F \not\models (\vee_{l \in q'} l)$$

- Can be rewritten as:

$$F \wedge \wedge_{l \in q} (\neg l) \models \bot \;\wedge\; \forall_{q' \subsetneq q} F \wedge \wedge_{l \in q} (\neg l) \not\models \bot$$

Reducing primes to group MUSes – prime implicates

- Recall definition of prime implicate $p \subseteq c$:

$$F \models (\vee_{l \in q} l) \ \wedge \ \forall_{q' \subsetneq q} F \not\models (\vee_{l \in q'} l)$$

- Can be rewritten as:

$$F \wedge \wedge_{l \in q}(\neg l) \models \bot \ \wedge \ \forall_{q' \subsetneq q} F \wedge \wedge_{l \in q}(\neg l) \not\models \bot$$

- Reduction: [BM07]
  - Start from implicate c
  - Formula F corresponds to background group G_0
  - Each literal l of c represents a group with a unit clause $(\neg l)$
  - Each group MUS represents prime implicate of F given c

Reducing primes to group MUSes – prime implicates

- Recall definition of prime implicate $p \subseteq c$:

$$F \vDash (\vee_{l \in q} l) \;\wedge\; \forall_{q' \subsetneq q} F \nvDash (\vee_{l \in q'} l)$$

- Can be rewritten as:

$$F \wedge \wedge_{l \in q} (\neg l) \vDash \bot \;\wedge\; \forall_{q' \subsetneq q} F \wedge \wedge_{l \in q} (\neg l) \nvDash \bot$$

- Reduction: [BM07]
  - Start from implicate c
  - Formula F corresponds to background group G_0
  - Each literal l of c represents a group with a unit clause $(\neg l)$
  - Each group MUS represents prime implicate of F given c

- **Note:** F is a (possibly non-clausal) propositional formula

How about prime implicants?

- Recall definition of prime implicant $p \subseteq t$:

$$(\wedge_{l \in p} l) \models F \ \wedge \ \forall_{p' \subsetneq p} (\wedge_{l \in p'} l) \not\models F$$

How about prime implicants?

- Recall definition of prime implicant $p \subseteq t$:

$$(\wedge_{l \in p} l) \models F \ \wedge \ \forall_{p' \subsetneq p} (\wedge_{l \in p'} l) \not\models F$$

- Can be rewritten as:

$$(\neg F) \wedge (\wedge_{l \in p} l) \models \bot \ \wedge \ \forall_{p' \subsetneq p} (\neg F) \wedge (\wedge_{l \in p'} l) \not\models \bot$$

How about prime implicants?

- Recall definition of prime implicant $p \subseteq t$:

$$(\wedge_{l \in p} l) \vDash F \ \wedge \ \forall_{p' \subsetneq p} \, (\wedge_{l \in p'} l) \nvDash F$$

- Can be rewritten as:

$$(\neg F) \wedge (\wedge_{l \in p} l) \vDash \bot \ \wedge \ \forall_{p' \subsetneq p} (\neg F) \wedge (\wedge_{l \in p'} l) \nvDash \bot$$

- Reduction: [BM07]
  - Start from implicant t
  - Formula $\neg F$ corresponds to background group G_0
  - Each literal l of t represents a group with a unit clause (l)
  - Each group MUS represents prime implicant of F given t

How about prime implicants?

- Recall definition of prime implicant $p \subseteq t$:

$$(\wedge_{l \in p} l) \vDash F \ \wedge \ \forall_{p' \subsetneq p} (\wedge_{l \in p'} l) \nvDash F$$

- Can be rewritten as:

$$(\neg F) \wedge (\wedge_{l \in p} l) \vDash \bot \ \wedge \ \forall_{p' \subsetneq p} (\neg F) \wedge (\wedge_{l \in p'} l) \nvDash \bot$$

- Reduction: [BM07]
  - Start from implicant t
  - Formula $\neg F$ corresponds to background group G_0
  - Each literal l of t represents a group with a unit clause (l)
  - Each group MUS represents prime implicant of F given t

- **How to compute group MUSes?**

Extracting MUSes

- Many algorithms, based on calls to SAT oracles:
  - Deletion-based [CD91,BDTW93]
  - QuickXplain [J04]
  - Progression [MSJB13]
  - ...

Extracting MUSes

- Many algorithms, based on calls to SAT oracles:
  - Deletion-based [CD91,BDTW93]
  - QuickXplain [J04]
  - Progression [MSJB13]
  - ...
- Several optimizations:
  - Clause set refinement [BDTW93,DHN06]
  - Recursive model rotation [BLMS12]
  - ...

Extracting MUSes

- Many algorithms, based on calls to SAT oracles:
  - Deletion-based [CD91,BDTW93]
  - QuickXplain [J04]
  - Progression [MSJB13]
  - ...
- Several optimizations:
  - Clause set refinement [BDTW93,DHN06]
  - Recursive model rotation [BLMS12]
  - ...
- Applicable to plain MUS or group MUS

Extracting MUSes

- Many algorithms, based on calls to SAT oracles:
  - Deletion-based [CD91,BDTW93]
  - QuickXplain [J04]
  - Progression [MSJB13]
  - ...
- Several optimizations:
  - Clause set refinement [BDTW93,DHN06]
  - Recursive model rotation [BLMS12]
  - ...
- Applicable to plain MUS or group MUS
- Applicable to computing primes

An example

$$F = (c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)$$

- Find prime implicate of F given implicate $(c \lor a)$

An example

$$F = (c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

- Find prime implicate of F given implicate $(c \vee a)$
- Group MUS formulation: $G_0 = F$; $G_1 = (\neg c)$; $G_2 = (\neg a)$

An example

$$F = (c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

- Find prime implicate of F given implicate $(c \vee a)$
- Group MUS formulation: $G_0 = F;\ G_1 = (\neg c);\ G_2 = (\neg a)$
- Standard deletion algorithm:

An example

$$F = (c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

- Find prime implicate of F given implicate $(c \vee a)$
- Group MUS formulation: $G_0 = F;\ G_1 = (\neg c);\ G_2 = (\neg a)$
- Standard deletion algorithm:
  - Drop $G_1 = (\neg c)$:

An example

$$F = (c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)$$

- Find prime implicate of F given implicate $(c \lor a)$
- Group MUS formulation: $G_0 = F;\ G_1 = (\neg c);\ G_2 = (\neg a)$
- Standard deletion algorithm:
  - Drop $G_1 = (\neg c)$:
    - $G_0 \land G_2 \nvDash \bot$, e.g. $c = b = 1$

An example

$$F = (c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

- Find prime implicate of F given implicate $(c \vee a)$
- Group MUS formulation: $G_0 = F;\ G_1 = (\neg c);\ G_2 = (\neg a)$
- Standard deletion algorithm:
  - Drop $G_1 = (\neg c)$:
    - $G_0 \wedge G_2 \not\models \perp$, e.g. $c = b = 1$
    - Thus, keep G_1

An example

$$F = (c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

- Find prime implicate of F given implicate $(c \vee a)$
- Group MUS formulation: $G_0 = F;\ G_1 = (\neg c);\ G_2 = (\neg a)$
- Standard deletion algorithm:
  - Drop $G_1 = (\neg c)$:
    - $G_0 \wedge G_2 \nvDash \perp$, e.g. $c = b = 1$
    - Thus, keep G_1
  - Drop $G_2 = (\neg a)$:

An example

$$F = (c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)$$

- Find prime implicate of F given implicate $(c \lor a)$
- Group MUS formulation: $G_0 = F;\ G_1 = (\neg c);\ G_2 = (\neg a)$
- Standard deletion algorithm:
  - Drop $G_1 = (\neg c)$:
    - $G_0 \land G_2 \nvDash \perp$, e.g. $c = b = 1$
    - Thus, keep G_1
  - Drop $G_2 = (\neg a)$:
    - $G_0 \land G_1 \vDash \perp$

An example

$$F = (c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

- Find prime implicate of F given implicate $(c \vee a)$
- Group MUS formulation: $G_0 = F$; $G_1 = (\neg c)$; $G_2 = (\neg a)$
- Standard deletion algorithm:
  - Drop $G_1 = (\neg c)$:
    - $G_0 \wedge G_2 \nvDash \perp$, e.g. $c = b = 1$
    - Thus, keep G_1
  - Drop $G_2 = (\neg a)$:
    - $G_0 \wedge G_1 \vDash \perp$
    - Thus, remove G_2

An example

$$F = (c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

- Find prime implicate of F given implicate $(c \vee a)$
- Group MUS formulation: $G_0 = F;\ G_1 = (\neg c);\ G_2 = (\neg a)$
- Standard deletion algorithm:
  - Drop $G_1 = (\neg c)$:
    - $G_0 \wedge G_2 \nvDash \perp$, e.g. $c = b = 1$
    - Thus, keep G_1
  - Drop $G_2 = (\neg a)$:
    - $G_0 \wedge G_1 \vDash \perp$
    - Thus, remove G_2
  - Group MUS: G_1

An example

$$F = (c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

- Find prime implicate of F given implicate $(c \vee a)$
- Group MUS formulation: $G_0 = F$; $G_1 = (\neg c)$; $G_2 = (\neg a)$
- Standard deletion algorithm:
  - Drop $G_1 = (\neg c)$:
    - $G_0 \wedge G_2 \nvDash \perp$, e.g. $c = b = 1$
    - Thus, keep G_1
  - Drop $G_2 = (\neg a)$:
    - $G_0 \wedge G_1 \vDash \perp$
    - Thus, remove G_2
  - Group MUS: G_1
  - $\{ \text{c} \}$ is a prime implicate of F, i.e. $F \vDash c$

Outline

Enumerating prime implicants of CNF formulae

- Search space must be larger than 2^n

Enumerating prime implicants of CNF formulae

- Search space must be larger than 2^n
- Work with modified formula H: [PPP99,JMSSS14]
  - Original variables: $\mathrm{var}(F) = \{v_1, \ldots, v_n\}$
  - Pair of new variables for each $v \in \mathrm{var}(F)$: $x_v, x_{\neg v}$

Enumerating prime implicants of CNF formulae

- Search space must be larger than 2^n
- Work with modified formula H: [PPP99,JMSSS14]
  - Original variables: $\mathrm{var}(F) = \{v_1, \ldots, v_n\}$
  - Pair of new variables for each $v \in \mathrm{var}(F)$: $x_v, x_{\neg v}$
  - Prevent one of the assignments to each new pair of variables:

$$L = \{(\neg x_v \vee \neg x_{\neg v}) \mid v \in \mathrm{var}(F)\}$$

Enumerating prime implicants of CNF formulae

- Search space must be larger than 2^n
- Work with modified formula H: [PPP99,JMSSS14]
  - Original variables: $\mathrm{var}(F) = \{v_1, \ldots, v_n\}$
  - Pair of new variables for each $v \in \mathrm{var}(F)$: $x_v, x_{\neg v}$
  - Prevent one of the assignments to each new pair of variables:

$$L = \{(\neg x_v \vee \neg x_{\neg v}) \mid v \in \mathrm{var}(F)\}$$

  - $x_v = x_{\neg v} = 0$: variable v unused
  - $x_v = 0 \wedge x_{\neg v} = 1$: negative literal of v used
  - $x_v = 1 \wedge x_{\neg v} = 0$: positive literal of v used

Enumerating prime implicants of CNF formulae

- Search space must be larger than 2^n
- Work with modified formula H: [PPP99,JMSSS14]
  - Original variables: $\mathrm{var}(F) = \{v_1, \ldots, v_n\}$
  - Pair of new variables for each $v \in \mathrm{var}(F)$: $x_v, x_{\neg v}$
  - Prevent one of the assignments to each new pair of variables:

$$L = \{(\neg x_v \vee \neg x_{\neg v}) \mid v \in \mathrm{var}(F)\}$$

    - $x_v = x_{\neg v} = 0$: variable v unused
    - $x_v = 0 \wedge x_{\neg v} = 1$: negative literal of v used
    - $x_v = 1 \wedge x_{\neg v} = 0$: positive literal of v used
  - Create C, by replacing each clause $c \in F$ with a new clause c_e:
    - For each $l \in c$, either add literal x_v, if $l = v$, or literal $x_{\neg v}$, if $l = \neg v$

Enumerating prime implicants of CNF formulae

- Search space must be larger than 2^n
- Work with modified formula H: [PPP99,JMSSS14]
  - Original variables: $\text{var}(F) = \{v_1, \ldots, v_n\}$
  - Pair of new variables for each $v \in \text{var}(F)$: $x_v, x_{\neg v}$
  - Prevent one of the assignments to each new pair of variables:

$$L = \{(\neg x_v \vee \neg x_{\neg v}) \mid v \in \text{var}(F)\}$$

    - $x_v = x_{\neg v} = 0$: variable v unused
    - $x_v = 0 \wedge x_{\neg v} = 1$: negative literal of v used
    - $x_v = 1 \wedge x_{\neg v} = 0$: positive literal of v used
  - Create C, by replacing each clause $c \in F$ with a new clause c_e:
    - For each $l \in c$, either add literal x_v, if $l = v$, or literal $x_{\neg v}$, if $l = \neg v$
  - Enumerate minimal models of $H = L \cup C$

Enumerating prime implicants of CNF formulae

- Search space must be larger than 2^n
- Work with modified formula H: [PPP99,JMSSS14]
  - Original variables: $\text{var}(F) = \{v_1, \ldots, v_n\}$
  - Pair of new variables for each $v \in \text{var}(F)$: $x_v, x_{\neg v}$
  - Prevent one of the assignments to each new pair of variables:

$$L = \{(\neg x_v \vee \neg x_{\neg v}) \,|\, v \in \text{var}(F)\}$$

    - $x_v = x_{\neg v} = 0$: variable v unused
    - $x_v = 0 \wedge x_{\neg v} = 1$: negative literal of v used
    - $x_v = 1 \wedge x_{\neg v} = 0$: positive literal of v used
  - Create C, by replacing each clause $c \in F$ with a new clause c_e:
    - For each $l \in c$, either add literal x_v, if $l = v$, or literal $x_{\neg v}$, if $l = \neg v$
  - Enumerate minimal models of $H = L \cup C$
- Use B (initially $B = \emptyset$) to block computed prime implicants
  - $H = L \cup C \cup B$

An example

$$F = (c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

An example

$$F = (c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)$$

- Define L:

$$L = (\neg x_a \lor \neg x_{\neg a}) \land (\neg x_b \lor \neg x_{\neg b}) \land (\neg x_c \lor \neg x_{\neg c}) \land (\neg x_d \lor \neg x_{\neg d})$$

An example

$$F = (c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

- Define L:

$$L = (\neg x_a \vee \neg x_{\neg a}) \wedge (\neg x_b \vee \neg x_{\neg b}) \wedge (\neg x_c \vee \neg x_{\neg c}) \wedge (\neg x_d \vee \neg x_{\neg d})$$

- Define C:

$$C = (x_c \vee x_a) \wedge (x_c \vee x_{\neg a}) \wedge (x_a \vee x_b \vee x_d) \wedge (x_a \vee x_b \vee x_{\neg d})$$

An example

$$F = (c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

- Define L:

$$L = (\neg x_a \vee \neg x_{\neg a}) \wedge (\neg x_b \vee \neg x_{\neg b}) \wedge (\neg x_c \vee \neg x_{\neg c}) \wedge (\neg x_d \vee \neg x_{\neg d})$$

- Define C:

$$C = (x_c \vee x_a) \wedge (x_c \vee x_{\neg a}) \wedge (x_a \vee x_b \vee x_d) \wedge (x_a \vee x_b \vee x_{\neg d})$$

- Let $H = L \cup C \cup B$

An example

$$F = (c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

- Define L:

$$L = (\neg x_a \vee \neg x_{\neg a}) \wedge (\neg x_b \vee \neg x_{\neg b}) \wedge (\neg x_c \vee \neg x_{\neg c}) \wedge (\neg x_d \vee \neg x_{\neg d})$$

- Define C:

$$C = (x_c \vee x_a) \wedge (x_c \vee x_{\neg a}) \wedge (x_a \vee x_b \vee x_d) \wedge (x_a \vee x_b \vee x_{\neg d})$$

- Let $H = L \cup C \cup B$
- Find minimal models:

An example

$$F = (c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

- Define L:

$$L = (\neg x_a \vee \neg x_{\neg a}) \wedge (\neg x_b \vee \neg x_{\neg b}) \wedge (\neg x_c \vee \neg x_{\neg c}) \wedge (\neg x_d \vee \neg x_{\neg d})$$

- Define C:

$$C = (x_c \vee x_a) \wedge (x_c \vee x_{\neg a}) \wedge (x_a \vee x_b \vee x_d) \wedge (x_a \vee x_b \vee x_{\neg d})$$

- Let $H = L \cup C \cup B$
- Find minimal models:
  - $x_b = x_c = 1$, i.e. prime implicant is $(b \wedge c)$; block with $(\neg x_b \vee \neg x_c)$

An example

$$F = (c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

- Define L:

$$L = (\neg x_a \vee \neg x_{\neg a}) \wedge (\neg x_b \vee \neg x_{\neg b}) \wedge (\neg x_c \vee \neg x_{\neg c}) \wedge (\neg x_d \vee \neg x_{\neg d})$$

- Define C:

$$C = (x_c \vee x_a) \wedge (x_c \vee x_{\neg a}) \wedge (x_a \vee x_b \vee x_d) \wedge (x_a \vee x_b \vee x_{\neg d})$$

- Let $H = L \cup C \cup B$
- Find minimal models:
  - $x_b = x_c = 1$, i.e. prime implicant is $(b \wedge c)$; block with $(\neg x_b \vee \neg x_c)$
  - $x_a = x_c = 1$, i.e. prime implicant is $(a \wedge c)$; block with $(\neg x_a \vee \neg x_c)$

An example

$$F = (c \vee a) \wedge (c \vee \neg a) \wedge (a \vee b \vee d) \wedge (a \vee b \vee \neg d)$$

- Define L:

$$L = (\neg x_a \vee \neg x_{\neg a}) \wedge (\neg x_b \vee \neg x_{\neg b}) \wedge (\neg x_c \vee \neg x_{\neg c}) \wedge (\neg x_d \vee \neg x_{\neg d})$$

- Define C:

$$C = (x_c \vee x_a) \wedge (x_c \vee x_{\neg a}) \wedge (x_a \vee x_b \vee x_d) \wedge (x_a \vee x_b \vee x_{\neg d})$$

- Let $H = L \cup C \cup B$
- Find minimal models:
  - $x_b = x_c = 1$, i.e. prime implicant is $(b \wedge c)$; block with $(\neg x_b \vee \neg x_c)$
  - $x_a = x_c = 1$, i.e. prime implicant is $(a \wedge c)$; block with $(\neg x_a \vee \neg x_c)$
  - No more (minimal) models

Other approaches

- Clausal formulae:
  - Problem reformulation
    - See above, but restricted
  - Iterated consensus/resolution, since the 1950s
  - Use of BDDs
    - ZRes [SdV'01]
    - ...
  - ...

Other approaches

- Clausal formulae:
  - Problem reformulation
    - See above, but restricted
  - Iterated consensus/resolution, since the 1950s
  - Use of BDDs
    - ZRes [SdV'01]
    - ...
  - ...
- Non-clausal formulae:
  - Use of BDDs
    - ZRes, with information about Tseitin variables [SdV'01]
    - ...
  - NNF, tries, etc.

Other approaches

- Clausal formulae:
  - Problem reformulation
    - See above, but restricted
  - Iterated consensus/resolution, since the 1950s
  - Use of BDDs
    - ZRes [SdV'01]
    - ...
  - ...
- Non-clausal formulae:
  - Use of BDDs
    - ZRes, with information about Tseitin variables [SdV'01]
    - ...
  - NNF, tries, etc.
  - Restricted to formulae with small number of variables

Outline

Background

Related Work

Primes for Non-Clausal Formulae

Results

An example

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

An example

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- Prime implicants of F?

An example

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- Prime implicants of F?
  - $(b \wedge c)$

An example

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- Prime implicants of F?
  - $(b \wedge c)$
  - $(a \wedge c)$
  - More?

An example

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- Prime implicants of F?
  - $(b \wedge c)$
  - $(a \wedge c)$
  - More?
- Prime implicates of F?

An example

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- Prime implicants of F?
  - $(b \wedge c)$
  - $(a \wedge c)$
  - More?
- Prime implicates of F?
  - (c)

An example

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- Prime implicants of F?
  - $(b \wedge c)$
  - $(a \wedge c)$
  - More?
- Prime implicates of F?
  - (c)
  - $(a \vee b)$
  - More?

An example

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- Prime implicants of F?
  - $(b \wedge c)$
  - $(a \wedge c)$
  - More?
- Prime implicates of F?
  - (c)
  - $(a \vee b)$
  - More?
- **How to enumerate primes of non-clausal formulae, with SAT oracles?**

Non-clausal prime compilation

- Recap SAT-based approach for CNF formulae:

$$H = L \cup C \cup B$$

Non-clausal prime compilation

- Recap SAT-based approach for CNF formulae:

$$H = L \cup C \cup B$$

  - L: Disallow $x_v = x_{\neg v} = 1$, for each pair $\{x_v, x_{\neg v}\}$
  - C: Encode clauses of F with new variables
  - B: Block computed prime implicants

Non-clausal prime compilation

- Recap SAT-based approach for CNF formulae:

$$H = L \cup C \cup B$$

  - L: Disallow $x_v = x_{\neg v} = 1$, for each pair $\{x_v, x_{\neg v}\}$
  - C: Encode clauses of F with new variables
  - B: Block computed prime implicants

- For non-clausal formulae, the problem is how to represent C, since F is not in CNF
  - Unrealistic to convert non-clausal formulae to CNF

Non-clausal prime compilation

- Recap SAT-based approach for CNF formulae:

$$H = L \cup C \cup B$$

  - L: Disallow $x_v = x_{\neg v} = 1$, for each pair $\{x_v, x_{\neg v}\}$
  - C: Encode clauses of F with new variables
  - B: Block computed prime implicants

- For non-clausal formulae, the problem is how to represent C, since F is not in CNF
  - Unrealistic to convert non-clausal formulae to CNF
  - And cannot introduce Tseitin variables
    - Primes not preserved

Non-clausal prime compilation

- Recap SAT-based approach for CNF formulae:

$$H = L \cup C \cup B$$

  - L: Disallow $x_v = x_{\neg v} = 1$, for each pair $\{x_v, x_{\neg v}\}$
  - C: Encode clauses of F with new variables
  - B: Block computed prime implicants

- For non-clausal formulae, the problem is how to represent C, since F is not in CNF
  - Unrealistic to convert non-clausal formulae to CNF
  - And cannot introduce Tseitin variables
    - Primes not preserved

- **Idea**: Construct C on demand as the algorithm executes; terminate when B blocks all primes **and** C equivalent to F

Non-clausal prime compilation – approach 1

- Iteratively compute maximal models A^H of working formula H
  - Initially $H = L; C = \emptyset; B = \emptyset$

Non-clausal prime compilation – approach 1

- Iteratively compute maximal models A^H of working formula H
  - Initially $H = L; C = \emptyset; B = \emptyset$
  - **Why maximal models?**

Non-clausal prime compilation – approach 1

- Iteratively compute maximal models A^H of working formula H
  - Initially $H = L; C = \emptyset; B = \emptyset$
  - **Why maximal models?**
    - Guarantees that one of the following two cases applies

Non-clausal prime compilation – approach 1

- Iteratively compute maximal models A^H of working formula H
  - Initially $H = L; C = \emptyset; B = \emptyset$
  - **Why maximal models?**
    - Guarantees that one of the following two cases applies
- Each maximal model A^H encodes assignment A^F to variables of F

Non-clausal prime compilation – approach 1

- Iteratively compute maximal models A^H of working formula H
  - Initially $H = L$; $C = \emptyset$; $B = \emptyset$
  - **Why maximal models?**
    - Guarantees that one of the following two cases applies

- Each maximal model A^H encodes assignment A^F to variables of F

- **Case 1**: If $A^F \vDash F$, then A^F is an implicant of F

Non-clausal prime compilation – approach 1

- Iteratively compute maximal models A^H of working formula H
  - Initially $H = L$; $C = \emptyset$; $B = \emptyset$
  - **Why maximal models?**
    - Guarantees that one of the following two cases applies
- Each maximal model A^H encodes assignment A^F to variables of F
- **Case 1**: If $A^F \models F$, then A^F is an implicant of F
  - Extract prime implicant
  - Report prime implicant
  - Block prime implicant (in B)

Non-clausal prime compilation – approach 1

- Iteratively compute maximal models A^H of working formula H
  - Initially $H = L; C = \emptyset; B = \emptyset$
  - **Why maximal models?**
    - Guarantees that one of the following two cases applies
- Each maximal model A^H encodes assignment A^F to variables of F
- **Case 1**: If $A^F \vDash F$, then A^F is an implicant of F
  - Extract prime implicant
  - Report prime implicant
  - Block prime implicant (in B)
- **Case 2**: If $F \vDash \neg A^F$, then A^F is an implicate of F

Non-clausal prime compilation – approach 1

- Iteratively compute maximal models A^H of working formula H
  - Initially $H = L; C = \emptyset; B = \emptyset$
  - **Why maximal models?**
    - Guarantees that one of the following two cases applies
- Each maximal model A^H encodes assignment A^F to variables of F
- **Case 1**: If $A^F \vDash F$, then A^F is an implicant of F
  - Extract prime implicant
  - Report prime implicant
  - Block prime implicant (in B)
- **Case 2**: If $F \vDash \neg A^F$, then A^F is an implicate of F
  - Extract prime implicate
  - Block prime implicate (in C)

Non-clausal prime compilation – approach 1

- Iteratively compute maximal models A^H of working formula H
  - Initially $H = L$; $C = \emptyset$; $B = \emptyset$
  - **Why maximal models?**
    - Guarantees that one of the following two cases applies

- Each maximal model A^H encodes assignment A^F to variables of F

- **Case 1**: If $A^F \vDash F$, then A^F is an implicant of F
  - Extract prime implicant
  - Report prime implicant
  - Block prime implicant (in B)

- **Case 2**: If $F \vDash \neg A^F$, then A^F is an implicate of F
  - Extract prime implicate
  - Block prime implicate (in C)

- Update H and repeat

Algorithm 1

input : Formula F
output: $PI_n(F)$ and prime implicate cover of F

$H \leftarrow \{(\neg x_v \vee \neg x_{\neg v}) \mid v \in \mathrm{var}(F)\}$ `# Initially,` $C = \emptyset$ `and` $B = \emptyset$

Algorithm 1

input : Formula F
output: $PI_n(F)$ and prime implicate cover of F

$H \leftarrow \{(\neg x_v \vee \neg x_{\neg v}) \mid v \in \mathrm{var}(F)\}$ `# Initially,` $C = \emptyset$ `and` $B = \emptyset$
while true **do**
  $(\mathrm{st}, A^H) \leftarrow \mathrm{MaxModel}(H)$
  if not st **then return**

Algorithm 1

input : Formula F
output: $PI_n(F)$ and prime implicate cover of F

$H \leftarrow \{(\neg x_v \vee \neg x_{\neg v}) \mid v \in \mathrm{var}(F)\}$ `# Initially,` $C = \emptyset$ `and` $B = \emptyset$
while true **do**
    $(\mathrm{st}, A^H) \leftarrow \mathrm{MaxModel}(H)$
    if not st **then return**
    $A^F \leftarrow \mathrm{Map}(A^H)$ `# Generate assignment for` F
    $\mathrm{st} \leftarrow \mathrm{SAT}(A^F \cup \neg F)$

Algorithm 1

input : Formula F
output: $PI_n(F)$ and prime implicate cover of F

```
H ← {(¬x_v ∨ ¬x_¬v) | v ∈ var(F)}        # Initially, C = ∅ and B = ∅
while true do
    (st, A^H) ← MaxModel(H)
    if not st then return
    A^F ← Map(A^H)                        # Generate assignment for F
    st ← SAT(A^F ∪ ¬F)
    if not st then                        # A^F ⊨ F; i.e. A^F is an implicant
        I_n ← ReduceImplicant(A^F, F)
        ReportPrimeImplicant(I_n)
        b ← {¬x_l | l ∈ I_n}              # Update B by blocking prime implicant
```

Algorithm 1

input : Formula F
output: $PI_n(F)$ and prime implicate cover of F

$H \leftarrow \{(\neg x_v \vee \neg x_{\neg v}) \mid v \in \mathrm{var}(F)\}$ # Initially, $C = \emptyset$ and $B = \emptyset$

while true **do**

  $(\mathrm{st}, A^H) \leftarrow \mathrm{MaxModel}(H)$

  if not st **then return**

  $A^F \leftarrow \mathrm{Map}(A^H)$ # Generate assignment for F

  $\mathrm{st} \leftarrow \mathrm{SAT}(A^F \cup \neg F)$

  if not st **then** # $A^F \vDash F$; i.e. A^F is an implicant

    $I_n \leftarrow \mathrm{ReduceImplicant}(A^F, F)$

    $\mathrm{ReportPrimeImplicant}(I_n)$

    $b \leftarrow \{\neg x_l \mid l \in I_n\}$ # Update B by blocking prime implicant

  else # $F \vDash \neg A^F$; i.e. $\neg A^F$ is an implicate

    $I_e \leftarrow \mathrm{ReduceImplicate}(A^F, F)$

    $b \leftarrow \{x_l \mid l \in I_e\}$ # Update C by blocking prime implicate

  $H \leftarrow H \cup \{b\}$

Example for algorithm 1

$$H = L \cup B \cup C$$

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- SAT oracle query: $F \wedge A^F$

A^H	A^F	Entailment	Update B/C
$x_a x_{\neg a} x_b x_{\neg b} x_c x_{\neg c}$			

Example for algorithm 1

$$H = L \cup B \cup C$$

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- SAT oracle query: $F \wedge A^F$

A^H $x_a x_{\neg a} x_b x_{\neg b} x_c x_{\neg c}$	A^F	Entailment	Update B/C
$A_1^H = 100101$	$A_1^F = a, \neg b, \neg c$	$F \models \neg A_1^F$	(x_c)

Example for algorithm 1

$$H = L \cup B \cup C$$

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- SAT oracle query: $F \wedge A^F$

A^H $x_a x_{\neg a} x_b x_{\neg b} x_c x_{\neg c}$	A^F	Entailment	Update B/C
$A_1^H = 100101$	$A_1^F = a, \neg b, \neg c$	$F \models \neg A_1^F$	(x_c)
$A_2^H = 100110$	$A_2^F = a, \neg b, c$	$A_2^F \models F$	$(\neg x_a \vee \neg x_c)$

Example for algorithm 1

$$H = L \cup B \cup C$$

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- SAT oracle query: $F \wedge A^F$

A^H $x_a x_{\neg a} x_b x_{\neg b} x_c x_{\neg c}$	A^F	Entailment	Update B/C
$A_1^H = 100101$	$A_1^F = a, \neg b, \neg c$	$F \models \neg A_1^F$	(x_c)
$A_2^H = 100110$	$A_2^F = a, \neg b, c$	$A_2^F \models F$	$(\neg x_a \vee \neg x_c)$
$A_3^H = 010110$	$A_3^F = \neg a, \neg b, c$	$F \models \neg A_3^F$	$(x_a \vee x_b)$

Example for algorithm 1

$$H = L \cup B \cup C$$

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- SAT oracle query: $F \wedge A^F$

A^H $x_a x_{\neg a} x_b x_{\neg b} x_c x_{\neg c}$	A^F	Entailment	Update B/C
$A_1^H = 100101$	$A_1^F = a, \neg b, \neg c$	$F \models \neg A_1^F$	(x_c)
$A_2^H = 100110$	$A_2^F = a, \neg b, c$	$A_2^F \models F$	$(\neg x_a \vee \neg x_c)$
$A_3^H = 010110$	$A_3^F = \neg a, \neg b, c$	$F \models \neg A_3^F$	$(x_a \vee x_b)$
$A_4^H = 011010$	$A_4^F = \neg a, b, c$	$A_4^F \models F$	$(\neg x_b \vee \neg x_c)$

Non-clausal prime compilation – approach 2

- Iteratively compute **minimal** models A^H of working formula H
  - Initially $H = L; C = \emptyset; B = \emptyset$

Non-clausal prime compilation – approach 2

- Iteratively compute **minimal** models A^H of working formula H
  - Initially $H = L$; $C = \emptyset$; $B = \emptyset$
  - **Why minimal models?**

Non-clausal prime compilation – approach 2

- Iteratively compute **minimal** models A^H of working formula H
  - Initially $H = L; C = \emptyset; B = \emptyset$
  - **Why minimal models?**
    - For prime implicants no need to reduce implicant

Non-clausal prime compilation – approach 2

- Iteratively compute **minimal** models A^H of working formula H
  - Initially $H = L$; $C = \emptyset$; $B = \emptyset$
  - **Why minimal models?**
    - For prime implicants no need to reduce implicant
- Each minimal model A^H encodes assignment A^F to variables of F

Non-clausal prime compilation – approach 2

- Iteratively compute **minimal** models A^H of working formula H
  - Initially $H = L$; $C = \emptyset$; $B = \emptyset$
  - **Why minimal models?**
    - For prime implicants no need to reduce implicant
- Each minimal model A^H encodes assignment A^F to variables of F
- If $A^F \models F$, then A^F is a **prime** implicant of F

Non-clausal prime compilation – approach 2

- Iteratively compute **minimal** models A^H of working formula H
  - Initially $H = L$; $C = \emptyset$; $B = \emptyset$
  - **Why minimal models?**
    - For prime implicants no need to reduce implicant
- Each minimal model A^H encodes assignment A^F to variables of F
- If $A^F \vDash F$, then A^F is a **prime** implicant of F
  - **No** need to extract prime implicant
  - Report prime implicant
  - Block prime implicant (in B)

Non-clausal prime compilation – approach 2

- Iteratively compute **minimal** models A^H of working formula H
  - Initially $H = L; C = \emptyset; B = \emptyset$
  - **Why minimal models?**
    - For prime implicants no need to reduce implicant
- Each minimal model A^H encodes assignment A^F to variables of F
- If $A^F \vDash F$, then A^F is a **prime** implicant of F
  - **No** need to extract prime implicant
  - Report prime implicant
  - Block prime implicant (in B)
- Else, find model $M^{\neg F}$ of $\neg F$, i.e. $M^{\neg F} \vDash \neg F$, and $\neg M^{\neg F}$ is an implicate of F

Non-clausal prime compilation – approach 2

- Iteratively compute **minimal** models A^H of working formula H
  - Initially $H = L$; $C = \emptyset$; $B = \emptyset$
  - **Why minimal models?**
    - For prime implicants no need to reduce implicant
- Each minimal model A^H encodes assignment A^F to variables of F
- If $A^F \vDash F$, then A^F is a **prime** implicant of F
  - **No** need to extract prime implicant
  - Report prime implicant
  - Block prime implicant (in B)
- Else, find model $M^{\neg F}$ of $\neg F$, i.e. $M^{\neg F} \vDash \neg F$, and $\neg M^{\neg F}$ is an implicate of F
  - Extract prime implicate
  - Block prime implicate (in C)

Non-clausal prime compilation – approach 2

- Iteratively compute **minimal** models A^H of working formula H
  - Initially $H = L$; $C = \emptyset$; $B = \emptyset$
  - **Why minimal models?**
    - For prime implicants no need to reduce implicant
- Each minimal model A^H encodes assignment A^F to variables of F
- If $A^F \vDash F$, then A^F is a **prime** implicant of F
  - **No** need to extract prime implicant
  - Report prime implicant
  - Block prime implicant (in B)
- Else, find model $M^{\neg F}$ of $\neg F$, i.e. $M^{\neg F} \vDash \neg F$, and $\neg M^{\neg F}$ is an implicate of F
  - Extract prime implicate
  - Block prime implicate (in C)
- Update H and repeat

Algorithm 2

input : Formula F
output: $PI_n(F)$ and prime implicate cover of F

$H \leftarrow \{(\neg x_v \vee \neg x_{\neg v}) \mid v \in \text{var}(F)\}$

Algorithm 2

input : Formula F
output: $PI_n(F)$ and prime implicate cover of F

$H \leftarrow \{(\neg x_v \vee \neg x_{\neg v}) \mid v \in \mathrm{var}(F)\}$
while true **do**
    $(\mathrm{st}, A^H) \leftarrow \mathrm{MinModel}(H)$
    if not st **then return**

Algorithm 2

input : Formula F
output: $PI_n(F)$ and prime implicate cover of F

$H \leftarrow \{(\neg x_v \vee \neg x_{\neg v}) \mid v \in \mathrm{var}(F)\}$
while true **do**
    $(\mathrm{st}, A^H) \leftarrow \mathrm{MinModel}(H)$
    if not st **then return**
    $A^F \leftarrow \mathrm{Map}(A^H)$
    $(\mathrm{st}, M^{\neg F}) \leftarrow \mathrm{SAT}(A^F \cup \neg F)$

Algorithm 2

input : Formula F
output: $PI_n(F)$ and prime implicate cover of F

$H \leftarrow \{(\neg x_v \vee \neg x_{\neg v}) \mid v \in \mathrm{var}(F)\}$
while true **do**
    $(\mathrm{st}, A^H) \leftarrow \mathrm{MinModel}(H)$
    if not st **then return**
    $A^F \leftarrow \mathrm{Map}(A^H)$
    $(\mathrm{st}, M^{\neg F}) \leftarrow \mathrm{SAT}(A^F \cup \neg F)$
    if st **then**   # $F \models \neg M^{\neg F}$; i.e. $\neg M^{\neg F}$ is an implicate
        $I_e \leftarrow \mathrm{ReduceImplicate}(M^{\neg F}, F)$
        $b \leftarrow \{x_l \mid l \in I_e\}$

Algorithm 2

input : Formula F
output: $PI_n(F)$ and prime implicate cover of F

$H \leftarrow \{(\neg x_v \vee \neg x_{\neg v}) \mid v \in \mathrm{var}(F)\}$
while true **do**
    $(\mathrm{st}, A^H) \leftarrow \mathrm{MinModel}(H)$
    if not st **then return**
    $A^F \leftarrow \mathrm{Map}(A^H)$
    $(\mathrm{st}, M^{\neg F}) \leftarrow \mathrm{SAT}(A^F \cup \neg F)$
    if st **then** # $F \vDash \neg M^{\neg F}$; i.e. $\neg M^{\neg F}$ is an implicate
        $I_e \leftarrow \mathrm{ReduceImplicate}(M^{\neg F}, F)$
        $b \leftarrow \{x_l \mid l \in I_e\}$
    else # $A^F \vDash F$; i.e. A^F is a prime implicant
        $I_n \leftarrow A^F$
        $\mathrm{ReportPrimeImplicant}(I_n)$
        $b \leftarrow \{\neg x_l \mid l \in I_n\}$
    $H \leftarrow H \cup \{b\}$

Example for algorithm 2

$$H = L \cup B \cup C$$

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- SAT oracle query: $F \wedge A^F$

A^H	A^F	$\neg M^{\neg F} / \neg \mathrm{st}$	B / C
$x_a x_{\neg a} x_b x_{\neg b} x_c x_{\neg c}$			

Example for algorithm 2

$$H = L \cup B \cup C$$

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- SAT oracle query: $F \wedge A^F$

A^H $x_a x_{\neg a} x_b x_{\neg b} x_c x_{\neg c}$	A^F	$\neg M^{\neg F} / \neg\text{st}$	B / C
000000	$A^F_1 = \emptyset$	$\neg a, \neg b, \neg c$	$(x_a \vee x_b)$

Example for algorithm 2

$$H = L \cup B \cup C$$

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- SAT oracle query: $F \wedge A^F$

A^H $x_a x_{\neg a} x_b x_{\neg b} x_c x_{\neg c}$	A^F	$\neg M^{\neg F} / \neg\text{st}$	B / C
000000	$A_1^F = \emptyset$	$\neg a, \neg b, \neg c$	$(x_a \vee x_b)$
001000	$A_2^F = b$	$\neg a, b, \neg c$	(x_c)

Example for algorithm 2

$$H = L \cup B \cup C$$

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- SAT oracle query: $F \wedge A^F$

A^H $x_a x_{\neg a} x_b x_{\neg b} x_c x_{\neg c}$	A^F	$\neg M^{\neg F} / \neg\mathrm{st}$	B / C
000000	$A_1^F = \emptyset$	$\neg a, \neg b, \neg c$	$(x_a \vee x_b)$
001000	$A_2^F = b$	$\neg a, b, \neg c$	(x_c)
001010	$A_3^F = b, c$	$\neg\mathrm{st}$	$(\neg x_b \vee \neg x_c)$

Example for algorithm 2

$$H = L \cup B \cup C$$

$$F = (((a \wedge b) \vee (a \wedge \neg b)) \wedge c) \vee (b \wedge c)$$

- SAT oracle query: $F \wedge A^F$

A^H $x_a x_{\neg a} x_b x_{\neg b} x_c x_{\neg c}$	A^F	$\neg M^{\neg F} / \neg \mathrm{st}$	B / C
000000	$A_1^F = \emptyset$	$\neg a, \neg b, \neg c$	$(x_a \vee x_b)$
001000	$A_2^F = b$	$\neg a, b, \neg c$	(x_c)
001010	$A_3^F = b, c$	$\neg \mathrm{st}$	$(\neg x_b \vee \neg x_c)$
100010	$A_4^F = a, c$	$\neg \mathrm{st}$	$(\neg x_a \vee \neg x_c)$

Outline

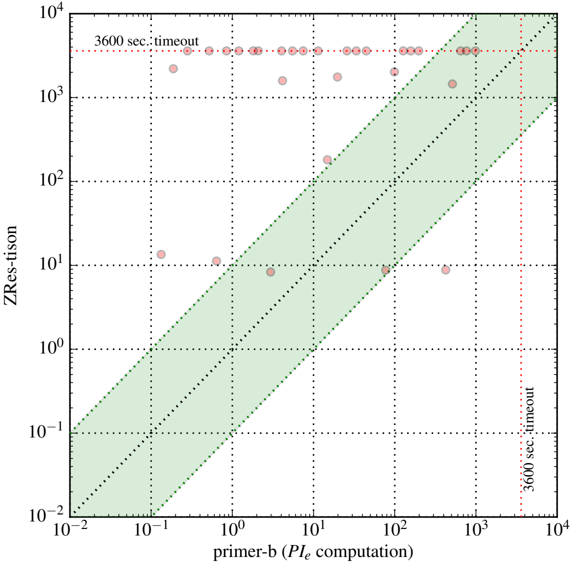
Experimental setup

- Server: Intel Xeon E5-2630 2.60GHz, 64GByte
- TO: 3600s
- MO: 10 GByte
- Tools:
  - primer: PRIMe compilER
  - zres-tison [SdV01]
- Benchmarks:
  - Quasigroup classification problems: 83
  - Cryptanalysis of the Geffe stream generator: 600
  - Crafted $F_m \vee PHP_n$: 30
    - $F_m = (x_1 \vee y_1) \wedge \cdots \wedge (x_m \vee y_m)$
    - $m \in \{10, \ldots, 20\}$
    - $n \in \{6, \ldots, 10\}$
  - Crafted $F_m \vee GT_n$: 30
    - $n \in \{12, \ldots, 20\}$

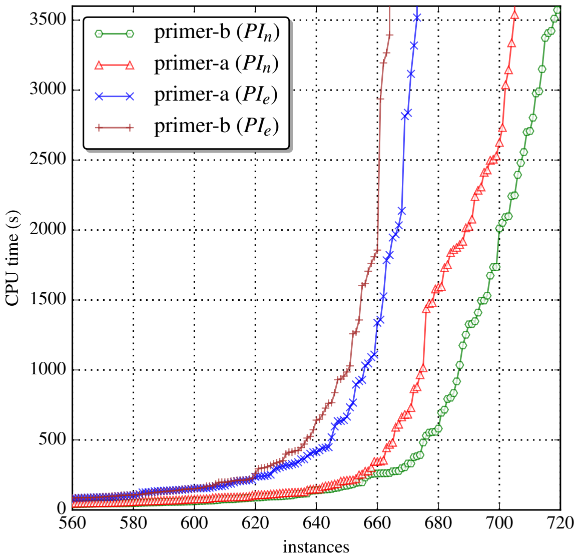
Summary of results

	QG6	Geffe gen.	F+PHP	F+GT	Total
# instances	83	600	30	30	743
ZRes-tison	0	0	11	0	11
primer-a (PI_n)	53	**596**	**30**	26	705
primer-a (PI_e)	28	588	**30**	27	673
primer-b (PI_n)	**64**	595	**30**	**30**	**719**
primer-b (PI_e)	30	577	**30**	27	664

F+PHP scatter plot



Comparing algorithms



Conclusions & future work

- Enumeration of prime implicants for non-clausal formulae with SAT oracles

Conclusions & future work

- Enumeration of prime implicants for non-clausal formulae with SAT oracles
  - Readily applicable to enumeration of prime implicates

Conclusions & future work

- Enumeration of prime implicants for non-clausal formulae with SAT oracles
  - Readily applicable to enumeration of prime implicates
  - Can be effective if number of primes is not too large

Conclusions & future work

- Enumeration of prime implicants for non-clausal formulae with SAT oracles
  - Readily applicable to enumeration of prime implicates
  - Can be effective if number of primes is not too large
  - Another instantiation of problem solving with SAT oracles

Conclusions & future work

- Enumeration of prime implicants for non-clausal formulae with SAT oracles
  - Readily applicable to enumeration of prime implicates
  - Can be effective if number of primes is not too large
  - Another instantiation of problem solving with SAT oracles
  - Exploiting recent work on computing MCSes (minimal/maximal models) and MUSes (prime implicants/implicates)
    - But also, MSMP in general

Conclusions & future work

- Enumeration of prime implicants for non-clausal formulae with SAT oracles
  - Readily applicable to enumeration of prime implicates
  - Can be effective if number of primes is not too large
  - Another instantiation of problem solving with SAT oracles
  - Exploiting recent work on computing MCSes (minimal/maximal models) and MUSes (prime implicants/implicates)
    - But also, MSMP in general
  - Another example of exploiting duality relationships in enumeration problems

Conclusions & future work

- Enumeration of prime implicants for non-clausal formulae with SAT oracles
  - Readily applicable to enumeration of prime implicates
  - Can be effective if number of primes is not too large
  - Another instantiation of problem solving with SAT oracles
  - Exploiting recent work on computing MCSes (minimal/maximal models) and MUSes (prime implicants/implicates)
    - But also, MSMP in general
  - Another example of exploiting duality relationships in enumeration problems
- Improvements to proposed algorithms

Conclusions & future work

- Enumeration of prime implicants for non-clausal formulae with SAT oracles
  - Readily applicable to enumeration of prime implicates
  - Can be effective if number of primes is not too large
  - Another instantiation of problem solving with SAT oracles
  - Exploiting recent work on computing MCSes (minimal/maximal models) and MUSes (prime implicants/implicates)
    - But also, MSMP in general
  - Another example of exploiting duality relationships in enumeration problems

- Improvements to proposed algorithms
- Applications of prime enumeration

Conclusions & future work

- Enumeration of prime implicants for non-clausal formulae with SAT oracles
  - Readily applicable to enumeration of prime implicates
  - Can be effective if number of primes is not too large
  - Another instantiation of problem solving with SAT oracles
  - Exploiting recent work on computing MCSes (minimal/maximal models) and MUSes (prime implicants/implicates)
    - But also, MSMP in general
  - Another example of exploiting duality relationships in enumeration problems

- Improvements to proposed algorithms
- Applications of prime enumeration
- Other compilation languages?

Thank You