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

\[ \text{[C71]} \]
The success of SAT

- Well-known NP-complete decision problem
- 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
- In practice, SAT is a success story of Computer Science
  - Hundreds (even more?) of practical applications
Problem solving with SAT oracles

- Decision Problems
- Counting Problems
- Function Problems
- Quantification Problems
- Enumeration Problems
Function problems

- Subset Minimal Sets
- Minimal Sets
- MaxSAT
- MinSAT
- PBO
- MUS
- MCS
- MSS
- ...
Function problems

- But also backbones, autarkies, MES, primes, etc.
An example – MUSes

\[(\bar{x}_1 \lor \bar{x}_2) (x_1) (x_5 \lor x_6) (\bar{x}_3 \lor \bar{x}_4) (x_2) (x_3) (x_4)\]

- Formula is unsatisfiable but **not** irreducible
An example – MUSes

\[(\bar{x}_1 \lor \bar{x}_2) (x_1) (x_5 \lor x_6) (\bar{x}_3 \lor \bar{x}_4) (x_2) (x_3) (x_4)\]

- Formula is **unsatisfiable** but **not** irreducible
- Can remove clauses, and formula still **unsatisfiable**
An example – MUSes

- 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

\[(\bar{x}_1 \lor \bar{x}_2) (x_1) (x_5 \lor x_6) (\bar{x}_3 \lor \bar{x}_4) (x_2) (x_3) (x_4)\]

- 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

\[(\bar{x}_1 \lor \bar{x}_2) \land (x_1) \land (x_5 \lor x_6) \land (\bar{x}_3 \lor \bar{x}_4) \land (x_2) \land (x_3) \land (x_4)\]

- 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

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

- Formula is unsatisfiable with satisfiable subformulas
An example – MCSes

- Formula is **unsatisfiable** with **satisfiable** subformulas
- Can remove clauses such that remaining clauses are **satisfiable**
An example – MCSes

- 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

\[(\bar{x}_1 \lor \bar{x}_2) (x_1) (x_5 \lor x_6) (\bar{x}_3 \lor \bar{x}_4) \]

\[(x_2) (x_3) (x_4)\]

- 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

\[(\bar{x}_1 \lor \bar{x}_2) \land (x_1) \land (x_5 \lor x_6) \land (\bar{x}_3 \lor \bar{x}_4) \land (x_2) \land (x_3) \land (x_4)\]

- 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

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

- 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

- Many applications: restore consistency; smallest MCSes are MaxSAT solutions; MUS enumeration; minimal/maximal models; etc.
Enumeration problems

- Model Enumeration
- MUS Enumeration
- MCS Enumeration
- ...

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

Quantification Problems

Σ₂^p

... 

FS₂^p

Σ_i^p, FΣ_i^p

Enumeration in the PH
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:
  
  \[(c \vee a) \land (c \vee \neg a) \land (a \vee b \vee d) \land (a \vee b \vee \neg d)\]

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

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

- How to enumerate primes of non-clausal formulae, with SAT oracles?
Application of enumeration – prime compilation

- Enumerate all prime implicates for:
  \[(c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)\]
  - Primes: \((c); (a \lor b)\)

- Enumerate all prime implicants for:
  \[(c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)\]

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

- How to enumerate primes of non-clausal formulae, with SAT oracles?
Application of enumeration – prime compilation

- Enumerate all prime implicates for:
  \[(c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)\]
  - Primes: \((c); (a \lor b)\)

- Enumerate all prime implicants for:
  \[(c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)\]
  - Primes: \((b \land c); (a \land c)\)
Application of enumeration – prime compilation

- Enumerate all prime implicates for:

  \[(c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)\]

  - Primes: \((c); (a \lor b)\)

- Enumerate all prime implicants for:

  \[(c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)\]

  - Primes: \((b \land c); (a \land c)\)

- Enumerate all prime implicants for:

  \[((((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c)\)
Application of enumeration – prime compilation

- Enumerate all prime implicates for:
  \[(c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)\]
  - Primes: \((c); (a \lor b)\)

- Enumerate all prime implicants for:
  \[((c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)\]
  - Primes: \((b \land c); (a \land c)\)

- Enumerate all prime implicants for:
  \[((((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c)\]
  - Primes: \((b \land c); (a \land c)\)
Application of enumeration – prime compilation

- **Enumerate all prime implicates for:**

\[(c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)\]

  - Primes: \((c); (a \lor b)\)

- **Enumerate all prime implicants for:**

\[(c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)\]

  - Primes: \((b \land c); (a \land c)\)

- **Enumerate all prime implicants for:**

\[(((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c)\]

  - Primes: \((b \land c); (a \land c)\)

- **Enumeration of primes studied since the 1930s!**

  - Formula minimization; Knowledge compilation; ...
Application of enumeration – prime compilation

• Enumerate all prime implicates for:

\[(c \vee a) \land (c \vee \neg a) \land (a \vee b \vee d) \land (a \vee b \vee \neg d)\]

  – Primes: \(c\); \((a \lor b)\)

• Enumerate all prime implicants for:

\[(c \vee a) \land (c \vee \neg a) \land (a \vee b \vee d) \land (a \vee b \vee \neg d)\]

  – Primes: \((b \land c)\); \((a \land c)\)

• Enumerate all prime implicants for:

\[((a \land b) \lor (a \land \neg b)) \land c\) \lor (b \land c)\]

  – Primes: \((b \land c)\); \((a \land c)\)

• Enumeration of primes studied since the 1930s!
  – Formula minimization; Knowledge compilation; ...

• How to enumerate primes of non-clausal formulae, with SAT oracles?
Outline

Background

Related Work

Primes for Non-Clausal Formulae

Results
Outline

Background

Related Work

Primes for Non-Clausal Formulae

Results
Propositional formulae

- Clausal:

- CNF: conjunction of disjunctions of literals

\[(c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)\]

- DNF: disjunction of conjunctions of literals

\[(c \land a) \lor (c \land \neg a) \lor (a \land b \land d) \lor (a \land b \land \neg d)\]

- 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

\[((a \land b) \lor (a \land \neg b)) \land c \lor (b \land c)\]
Propositional formulae

- **Clausal:**
  - **CNF:** conjunction of disjunctions of literals

\[(c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)\]
Propositional formulae

• Clausal:
  – **CNF**: conjunction of disjunctions of literals
    
    \[(c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)\]

  – **DNF**: disjunction of conjunctions of literals
    
    \[(c \land a) \lor (c \land \neg a) \lor (a \land b \land d) \lor (a \land b \land \neg d)\]
Propositional formulae

- **Clausal:**
  - **CNF:** conjunction of disjunctions of literals
    \[(c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)\]
  - **DNF:** disjunction of conjunctions of literals
    \[(c \land a) \lor (c \land \neg a) \lor (a \land b \land d) \lor (a \land b \land \neg d)\]
  - Other notation: **Product of Sums (POS)** / **Sum of Products (SOP)**
Propositional formulae

- **Clausal:**
  - **CNF:** conjunction of disjunctions of literals
    \[(c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d)\]
  - **DNF:** disjunction of conjunctions of literals
    \[(c \land a) \lor (c \land \neg a) \lor (a \land b \land d) \lor (a \land b \land \neg d)\]
  - 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 \(\neg, \land, \lor\)

\[((((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c))\]
Defining primes

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

  $$F \models (\lor_{l \in q} l) \land \forall q' \subset q \ F \not\models (\lor_{l \in q'} l)$$

- 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.

$$F \models (\forall l \in q) \land \forall q' \subset q \ F \not\models (\forall l \in q')$$

- 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.

$$(\land l \in p) \models F \land \forall p' \subset p (\land l \in p') \not\models F$$

- 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.

$$F \models (\lor_{l \in q} l) \land \forall q \subseteq q' F \not\models (\lor_{l \in q'} l)$$

• 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.

$$(\land_{l \in p} l) \models F \land \forall p' \subseteq p (\land_{l \in p'} l) \not\models F$$

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

• Each prime implicant (resp. implicate) of $F$ is a minimal hitting set of the prime impicates (resp. implicants) of $F$ [R94]
Computing primes

- Extract one prime implicant for $F$ in CNF:

  - Find satisfying assignment $\mu$ of $F$
  - Drop literals from $\mu$ 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...
Computing primes

• Extract one prime implicant for $F$ in CNF:
  – Find satisfying assignment $\mu$ of $F$
Computing primes

- Extract one prime implicant for $F$ in CNF:
  - Find satisfying assignment $\mu$ of $F$
  - Drop literals from $\mu$ 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...
Computing primes

• Extract one prime implicant for $F$ in CNF:
  – Find satisfying assignment $\mu$ of $F$
  – Drop literals from $\mu$ 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 \( \mu \) of \( F \)
  – Drop literals from \( \mu \) 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 $F \models \bot$:

  - $M \subseteq F$ is a Minimal Unsatisfiable Subset (MUS) iff:
    
    $M \models \bot \land \forall M' \subset M, M' \not\models \bot$

  - $S \subseteq F$ is a Maximal Satisfiable Subset (MSS) iff:
    
    $S \not\models \bot \land \forall S' \subseteq S, S' \not\models \bot$

  - $C \subseteq F$ is a Minimal Correction Subset (MCS) iff:
    
    $F \setminus C \not\models \bot \land \forall C' \subset C, F \setminus C' \not\models \bot$

- 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]
Defining MUSes/MCSes/MSSes

- Given CNF $F$, with $F \models \bot$:
  - $M \subseteq F$ is a Minimal Unsatisfiable Subset (MUS) iff:
    $$M \models \bot \land \forall_{M' \subsetneq M} M' \not\models \bot$$
  - $S \subseteq F$ is a Maximal Satisfiable Subset (MSS) iff:
    $$S \not\models \bot \land \forall_{S' \supseteq S} S' \models \bot$$
  - $C \subseteq F$ is a Minimal Correction Subset (MCS) iff:
    $$F \setminus C \not\models \bot \land \forall_{C' \supsetneq C} F \setminus C' \not\models \bot$$

- 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]
Defining MUSes/MCSes/MSSes

- Given CNF $F$, with $F \models \bot$:
  - $M \subseteq F$ is a Minimal Unsatisfiable Subset (MUS) iff:
    $$M \models \bot \land \forall_{M' \subseteq M} M' \not\models \bot$$
  - $S \subseteq F$ is a Maximal Satisfiable Subset (MSS) iff:
    $$S \not\models \bot \land \forall_{S' \subseteq F} S' \models \bot$$

- 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 $F$.

References: [R'87, BL'03, BS'05, LS'08]
Defining MUSes/MCSes/MSSes

- Given CNF $F$, with $F \models \bot$:
  - $M \subseteq F$ is a Minimal Unsatisfiable Subset (MUS) iff:
    $$M \models \bot \land \forall_{M' \subsetneq M} M' \not\models \bot$$
  - $S \subseteq F$ is a Maximal Satisfiable Subset (MSS) iff:
    $$S \not\models \bot \land \forall_{S' \subseteq F} S' \models \bot$$
  - $C \subseteq F$ is a Minimal Correction Subset (MCS) iff:
    $$F \setminus C \not\models \bot \land \forall_{C' \subsetneq C} F \setminus C' \models \bot$$
Defining MUSes/MCSes/MSSes

- Given CNF $F$, with $F \models \bot$:
  - $M \subseteq F$ is a Minimal Unsatisfiable Subset (MUS) iff:
    $$M \models \bot \land \forall M' \subsetneq M \ M' \not\models \bot$$
  - $S \subseteq F$ is a Maximal Satisfiable Subset (MSS) iff:
    $$S \not\models \bot \land \forall S \subsetneq S' \subseteq F \ S' \models \bot$$
  - $C \subseteq F$ is a Minimal Correction Subset (MCS) iff:
    $$F \setminus C \not\models \bot \land \forall C' \subsetneq C \ F \setminus C' \models \bot$$
  - 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 $F \models \bot$:
  - $M \subseteq F$ is a Minimal Unsatisfiable Subset (MUS) iff:
    $$M \models \bot \land \forall M' \subsetneq M \, M' \not\models \bot$$
  - $S \subseteq F$ is a Maximal Satisfiable Subset (MSS) iff:
    $$S \not\models \bot \land \forall S \subseteq S' \subseteq F \, S' \models \bot$$
  - $C \subseteq F$ is a Minimal Correction Subset (MCS) iff:
    $$F \setminus C \not\models \bot \land \forall C' \subsetneq C \, F \setminus C' \models \bot$$

- 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 $\Gamma = \{G_1, \ldots, G_k\}$
• 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:

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

$$\forall \Psi' \subset \Psi \bigwedge_{G_i \in G_0 \cup \Psi} (c) \not\models \bot$$
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:

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

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

$$\bigwedge_{G_i \in G_0 \cup \Psi} \bigwedge_{c \in G_i} (c) \models \bot \land \forall \Psi' \not\subseteq \Psi \bigwedge_{G_i \in G_0 \cup \Psi'} (c) \not\models \bot$$
Reducing primes to group MUSes – prime implicates

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

$$F \models (\lor l \in q) \land \forall q' \subsetneq q F \not\models (\lor l \in q')$$

- Reduction:
  - 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
Reducing primes to group MUSes – prime implicants

- Recall definition of prime implicate \( p \subseteq c \):

\[
F \models (\lor_{l \in q} l) \land \forall_{q' \subset q} F \not\models (\lor_{l \in q'} l)
\]

- Can be rewritten as:

\[
F \land \land_{l \in q} (\neg l) \models \bot \land \forall_{q' \subset q} F \land \land_{l \in q} (\neg l) \not\models \bot
\]
Reducing primes to group MUSes – prime implicates

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

$$F \models (\lor l \in q) \land \forall q' \subset q F \not\models (\lor l \in q')$$

- Can be rewritten as:

$$F \land \land l \in q (\neg l) \models \bot \land \forall q' \subset q F \land \land l \in q (\neg l) \not\models \bot$$

- Reduction:
  - 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 $c$

[BM07]
Reducing primes to group MUSes – prime implicates

- Recall definition of prime implicate \( p \subseteq c \):

\[
F \models (\forall l \in q l) \land \forall q' \subsetneq q F \not\models (\forall l \in q' l)
\]

- Can be rewritten as:

\[
F \land \land l \in q (\neg l) \models \bot \land \forall q' \subsetneq q F \land \land l \in q (\neg l) \not\models \bot
\]

- Reduction:

  - 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 $p \subseteq t$:

$$\left(\bigwedge_{l \in p} l\right) \models F \land \forall p' \subseteq p \left(\bigwedge_{l \in p'} l\right) \not\models F$$

- Can be rewritten as:

$$\left(\neg F\right) \land \left(\bigwedge_{l \in l} l\right) \models \bot \land \forall p' \subseteq p \left(\neg F \land \bigwedge_{l \in p'} l\right) \not\models \bot$$

- 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?
How about prime implicants?

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

$$\left( \land_{l \in p} l \right) \models F \land \forall_{p' \subsetneq p} \left( \land_{l \in p'} l \right) \not\models F$$

- Can be rewritten as:

$$\left( \neg F \right) \land \left( \land_{l \in p} l \right) \models \bot \land \forall_{p' \subsetneq p} \left( \neg F \right) \land \left( \land_{l \in p'} l \right) \not\models \bot$$
How about prime implicants?

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

$$(\wedge_{l \in p} l) \models F \land \forall_{p' \subset p} (\wedge_{l \in p'} l) \not\models F$$

• Can be rewritten as:

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

• Reduction:

  - 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 \( p \subseteq t \):
  \[
  (\land_{l \in p} l) \models F \land \forall_{p' \subset p} (\land_{l \in p'} l) \not\models F
  \]

- Can be rewritten as:
  \[
  (\neg F) \land (\land_{l \in p} l) \models \bot \land \forall_{p' \subset p} (\neg F) \land (\land_{l \in p'} l) \not\models \bot
  \]

- Reduction:
  - 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
  - QuickXplain
  - Progression
  - ...

[CD91, BDTW93]
[Jo04]
[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
  – QuickXplain
  – Progression
  – ...

• Several optimizations:
  – Clause set refinement
  – Recursive model rotation
  – ...

• Applicable to plain MUS or group MUS
Extracting MUSes

- Many algorithms, based on calls to SAT oracles:
  - Deletion-based
  - QuickXplain
  - Progression
  - ...

- Several optimizations:
  - Clause set refinement
  - Recursive model rotation
  - ...

- 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

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

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

Thus, keep \( G_1 \) – Drop \( G_2 = (\neg a) \):

Thus, remove \( G_2 \) – Group MUS: \( G_1 \) – \{c\} is a prime implicate of \( F \), i.e. \( F \models c \)
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 \( G_1 = (\neg c) \):
    - \( G_0 \land G_2 \neq \bot \), e.g. \( c = b = 1 \)
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 \( G_1 = (\neg c) \):
    - \( G_0 \land G_2 \not\models \bot \), e.g. \( c = b = 1 \)
    - Thus, keep \( G_1 \)
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 \( G_1 = \neg c \):
    - \( G_0 \land G_2 \neq \bot \), 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 \( G_1 = (\neg c) \):
    - \( G_0 \land G_2 \not\models \bot \), e.g. \( c = b = 1 \)
    - Thus, keep \( G_1 \)
  - Drop \( G_2 = (\neg a) \):
    - \( G_0 \land G_1 \models \bot \)
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 \( G_1 = (\neg c) \):
    - \( G_0 \land G_2 \neq \bot \), e.g. \( c = b = 1 \)
    - Thus, keep \( G_1 \)
  - Drop \( G_2 = (\neg a) \):
    - \( G_0 \land G_1 \models \bot \)
    - Thus, remove \( G_2 \)
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 \( G_1 = (\neg c) \):
    ▶ \( G_0 \land G_2 \neq \bot \), e.g. \( c = b = 1 \)
    ▶ Thus, keep \( G_1 \)
  - Drop \( G_2 = (\neg a) \):
    ▶ \( G_0 \land G_1 \models \bot \)
    ▶ Thus, remove \( G_2 \)
  - Group MUS: \( G_1 \)
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 \( G_1 = (\neg c) \):
    ▶ \( G_0 \land G_2 \not\models \bot \), e.g. \( c = b = 1 \)
    ▶ Thus, keep \( G_1 \)
  – Drop \( G_2 = (\neg a) \):
    ▶ \( G_0 \land G_1 \models \bot \)
    ▶ Thus, remove \( G_2 \)
  – Group MUS: \( G_1 \)
  – \( \{ c \} \) is a prime implicate of \( F \), i.e. \( F \models c \)
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: $\text{var}(F) = \{v_1, \ldots, v_n\}$
  - Pair of new variables for each $v \in \text{var}(F)$: $x_v, x_{\neg v}$
Enumerating prime implicants of CNF formulae

- Search space must be larger than $2^n$
- Work with modified formula $H$:
  - 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:
    \[ L = \{(-x_v \lor -x_{\neg v}) \mid v \in \text{var}(F)\} \]
Enumerating prime implicants of CNF formulae

- Search space must be larger than $2^n$

- Work with modified formula $H$:
  
  - 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:

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

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

[PPP99, JMSSS14]
Enumerating prime implicants of CNF formulae

- Search space must be larger than $2^n$
- Work with modified formula $H$:
  - 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:
    
    $$L = \{(\neg x_v \lor \neg x_{\neg v}) \mid v \in \text{var}(F)\}$$

    - $x_v = x_{\neg v} = 0$: variable $v$ unused
    - $x_v = 0 \land x_{\neg v} = 1$: negative literal of $v$ used
    - $x_v = 1 \land x_{\neg v} = 0$: 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 $H$:
  - 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:
    
    $$L = \{(\neg x_v \lor \neg x_{\neg v}) \mid v \in \text{var}(F)\}$$
  
    - $x_v = x_{\neg v} = 0$: variable $v$ unused
    - $x_v = 0 \land x_{\neg v} = 1$: negative literal of $v$ used
    - $x_v = 1 \land x_{\neg v} = 0$: 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 $H = L \cup C$

[PPP99, JMSSS14]
Enumerating prime implicants of CNF formulae

- Search space must be larger than $2^n$
- Work with modified formula $H$:
  - 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:
    $$L = \{ (\neg x_v \lor \neg x_{\neg v}) \mid v \in \text{var}(F) \}$$
    - $x_v = x_{\neg v} = 0$: variable $v$ unused
    - $x_v = 0 \land x_{\neg v} = 1$: negative literal of $v$ used
    - $x_v = 1 \land x_{\neg v} = 0$: 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

\[ F = (c \lor a) \land (c \lor \neg a) \land (a \lor b \lor d) \land (a \lor b \lor \neg d) \]
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 $L$:

$$L = (\neg x_a \lor \neg \neg x_a) \land (\neg x_b \lor \neg \neg x_b) \land (\neg x_c \lor \neg \neg x_c) \land (\neg x_d \lor \neg \neg x_d)$$
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 \( L \):

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

• Define \( C \):

\[ C = (x_c \lor x_a) \land (x_c \lor x_{\neg a}) \land (x_a \lor x_b \lor x_d) \land (x_a \lor x_b \lor x_{\neg d}) \]
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 \( L \):
  \[ L = (\neg x_a \lor \neg x_{\neg a}) \land (\neg x_b \lor \neg x_{\neg b}) \land (\neg x_c \lor \neg x_{\neg c}) \land (\neg x_d \lor \neg x_{\neg d}) \]

- Define \( C \):
  \[ C = (x_c \lor x_a) \land (x_c \lor x_{\neg a}) \land (x_a \lor x_b \lor x_d) \land (x_a \lor x_b \lor x_{\neg d}) \]

- Let \( H = L \cup C \cup B \)
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 \( L \):
  \[ L = (\neg x_a \lor \neg \neg x_a) \land (\neg x_b \lor \neg \neg x_b) \land (\neg x_c \lor \neg \neg x_c) \land (\neg x_d \lor \neg \neg x_d) \]

- Define \( C \):
  \[ C = (x_c \lor x_a) \land (x_c \lor \neg x_a) \land (x_a \lor x_b \lor x_d) \land (x_a \lor x_b \lor \neg x_d) \]

- Let \( H = L \cup C \cup B \)

- Find minimal models:
An example

\[ F = (c \lor a) \land (c \lor \lnot a) \land (a \lor b \lor d) \land (a \lor b \lor \lnot d) \]

- Define \( L \):

\[ L = (\lnot x_a \lor \lnot x_{\lnot a}) \land (\lnot x_b \lor \lnot x_{\lnot b}) \land (\lnot x_c \lor \lnot x_{\lnot c}) \land (\lnot x_d \lor \lnot x_{\lnot d}) \]

- Define \( C \):

\[ C = (x_c \lor x_a) \land (x_c \lor x_{\lnot a}) \land (x_a \lor x_b \lor x_d) \land (x_a \lor x_b \lor x_{\lnot d}) \]

- Let \( H = L \cup C \cup B \)

- Find minimal models:
  - \( x_b = x_c = 1 \), i.e. prime implicant is \( (b \land c) \); block with \( (\lnot x_b \lor \lnot x_c) \)
An example

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

- Define \( L \):
  \[ L = (\neg x_a \lor \neg x_{\neg a}) \land (\neg x_b \lor \neg x_{\neg b}) \land (\neg x_c \lor \neg x_{\neg c}) \land (\neg x_d \lor \neg x_{\neg d}) \]

- Define \( C \):
  \[ C = (x_c \lor x_a) \land (x_c \lor x_{\neg a}) \land (x_a \lor x_b \lor x_d) \land (x_a \lor x_b \lor x_{\neg d}) \]

- Let \( H = L \cup C \cup B \)

- Find minimal models:
  - \( x_b = x_c = 1 \), i.e. prime implicant is \((b \land c)\); block with \((\neg x_b \lor \neg x_c)\)
  - \( x_a = x_c = 1 \), i.e. prime implicant is \((a \land c)\); block with \((\neg x_a \lor \neg x_c)\)
An example

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

- Define \( L \):
  \[ L = (\neg x_a \lor \neg x_{\neg a}) \land (\neg x_b \lor \neg x_{\neg b}) \land (\neg x_c \lor \neg x_{\neg c}) \land (\neg x_d \lor \neg x_{\neg d}) \]

- Define \( C \):
  \[ C = (x_c \lor x_a) \land (x_c \lor x_{\neg a}) \land (x_a \lor x_b \lor x_d) \land (x_a \lor x_b \lor x_{\neg d}) \]

- Let \( H = L \cup C \cup B \)

- Find minimal models:
  - \( x_b = x_c = 1 \), i.e. prime implicant is \((b \land c)\); block with \((\neg x_b \lor \neg x_c)\)
  - \( x_a = x_c = 1 \), i.e. prime implicant is \((a \land c)\); block with \((\neg x_a \lor \neg x_c)\)
  - No more (minimal) models
Other approaches

- **Clausal formulae:**
  - Problem reformulation
    - See above, but restricted
  - Iterated consensus/resolution, since the 1950s
  - Use of BDDs
    - ZRes
    - ...
Other approaches

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

- **Non-clausal formulae:**
  - Use of **BDDs**
    - ZRes, with information about Tseitin variables
    - ...
  - NNF, tries, etc.

[SdV'01]
Other approaches

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

- **Non-clausal formulae:**
  - Use of BDDs
    - ZRes, with information about Tseitin variables
    - ...
  - NNF, tries, etc.
  - Restricted to formulae with small number of variables

[SdV'01]
Outline

Background

Related Work

Primes for Non-Clausal Formulae

Results
An example

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]
An example

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]

- Prime implicants of \( F \)?
An example

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]

- Prime implicants of \( F \)?
  - \((b \land c)\)
An example

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]

- Prime implicants of \( F \)?
  - \((b \land c)\)
  - \((a \land c)\)
  - More?
An example

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]

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

- Prime implicates of \( F \)?
An example

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]

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

- Prime implicates of \( F \)?
  - \((c)\)
An example

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

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

- Prime implicants of $F$?
  - $(c)$
  - $(a \lor b)$
  - More?
An example

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]

• Prime implicants of \( F \)?
  - \( (b \land c) \)
  - \( (a \land c) \)
  - More?

• Prime implicates of \( F \)?
  - \( (c) \)
  - \( (a \lor b) \)
  - More?

• How to enumerate primes of non-clausal formulae, with SAT oracles?
Non-clausal prime compilation

- Recap SAT-based approach for CNF formulae:

\[ H = L \cup C \cup B \]

- 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

- 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

- 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

• 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 $A^F \models F$, 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 $A^F \models F$, then $A^F$ is an implicant of $F$
  – Extract prime implicant
  – Report prime implicant
  – Block prime implicant (in $B$)

• **Case 2**: If $F \models \neg A^F$, 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 $A^F \models F$, then $A^F$ is an implicant of $F$
  – Extract prime implicant
  – Report prime implicant
  – Block prime implicant (in $B$)

• **Case 2:** If $F \models \neg A^F$, 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 $A^F \models F$, then $A^F$ is an implicant of $F$
  - Extract prime implicant
  - Report prime implicant
  - Block prime implicant (in $B$)

• Case 2: If $F \models \neg A^F$, 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**: $Pl_n(F)$ and prime implicate cover of $F$

\[ H \leftarrow \{\neg x_v \lor \neg x_{\neg v} \mid v \in \text{var}(F)\} \quad \# \text{Initially, } C = \emptyset \text{ and } B = \emptyset \]
Algorithm 1

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

\[ H \leftarrow \{ (\neg x_v \lor \neg x_{\neg v}) \mid v \in \text{var}(F) \} \quad \# \text{Initially, } C = \emptyset \text{ and } B = \emptyset \]

while true do
  \[(\text{st, } A^H) \leftarrow \text{MaxModel}(H)\]
  if not st then return

\[ b \leftarrow \{ \neg x_l \mid l \in I_n \} \quad \# \text{Update } B \text{ by blocking prime implicant} \]
\[ I_n \leftarrow \text{ReduceImplicant}(A^H, F) \]
\[ \text{ReportPrimeImplicant}(I_n) \]

\[ b \leftarrow \{ x_l \mid l \in I_e \} \quad \# \text{Update } C \text{ by blocking prime implicate} \]
\[ H \leftarrow H \cup \{ b \} \]
Algorithm 1

**input** : Formula $F$

**output**: $Pl_n(F)$ and prime implicate cover of $F$

$H \leftarrow \{ (\neg x_v \lor \neg x_{\neg v}) \mid v \in \text{var}(F) \}$ \hspace{1cm} \# Initially, $C = \emptyset$ and $B = \emptyset$

**while** true **do**

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

**if** not **st** **then return**

$A^F \leftarrow \text{Map}(A^H)$ \hspace{1cm} \# Generate assignment for $F$

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

\( b \leftarrow \{ \neg x_l \mid l \in I_n \} \)

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

**ReportPrimeImplicant**($I_n$)

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

**else**

\( b \leftarrow \{ x_l \mid l \in I_e \} \)

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

$B \leftarrow \{ x_l \mid l \in I_e \}$

$C \leftarrow \{ \neg x_l \mid l \in I_e \}$

$H \leftarrow H \cup \{ b \}$
Algorithm 1

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

$H \leftarrow \{(\neg x_v \lor \neg x_{\neg v}) \mid v \in \text{var}(F)\}$  \hspace{1cm} # Initially, $C = \emptyset$ and $B = \emptyset$

while true do
    $(\text{st}, A^H) \leftarrow \text{MaxModel}(H)$
    if not st then return
    $A^F \leftarrow \text{Map}(A^H)$  \hspace{1cm} # Generate assignment for $F$
    $\text{st} \leftarrow \text{SAT}(A^F \cup \neg F)$
    if not st then  \hspace{1cm} # $A^F \models F$; i.e. $A^F$ is an implicant
        $I_n \leftarrow \text{ReduceImplicant}(A^F, F)$
        ReportPrimeImplicant($I_n$)
        $b \leftarrow \{\neg x_l \mid l \in I_n\}$  \hspace{1cm} # Update $B$ by blocking prime implicant
        $H \leftarrow H \cup \{b\}$
Algorithm 1

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

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

while true do
    $(st, A^H) \leftarrow \text{MaxModel}(H)$
    if not $st$ then return
    $A^F \leftarrow \text{Map}(A^H)$  # Generate assignment for $F$
    $st \leftarrow \text{SAT}(A^F \cup \neg F)$
    if not $st$ then  # $A^F \models F$; i.e. $A^F$ is an implicant
        $I_n \leftarrow \text{ReduceImplicant}(A^F, F)$
        ReportPrimeImplicant($I_n$)
        $b \leftarrow \{\neg x_l \mid l \in I_n\}$  # Update $B$ by blocking prime implicant
    else  # $F \models \neg A^F$; i.e. $\neg A^F$ is an implicate
        $I_e \leftarrow \text{ReduceImplicate}(A^F, F)$
        $b \leftarrow \{x_l \mid l \in I_e\}$  # Update $C$ by blocking prime implicate
        $H \leftarrow H \cup \{b\}$
Example for algorithm 1

\[ H = L \cup B \cup C \]

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]

- SAT oracle query: \( F \land A^F \)

<table>
<thead>
<tr>
<th>( A^H )</th>
<th>( A^F )</th>
<th>Entailment</th>
<th>Update ( B/C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_a \land \neg a \land x_b \land \neg b \land x_c \land \neg c )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example for algorithm 1

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

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

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

<table>
<thead>
<tr>
<th>$A^H$</th>
<th>$A^F$</th>
<th>Entailment</th>
<th>Update $B/C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_a \neg a x_b \neg b x_c \neg c$</td>
<td>$A^F_1 = a, \neg b, \neg c$</td>
<td>$F \models \neg A^F_1$</td>
<td>$(x_c)$</td>
</tr>
</tbody>
</table>
Example for algorithm 1

\[ H = L \cup B \cup C \]

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]

- SAT oracle query: \( F \land A^F \)

<table>
<thead>
<tr>
<th>( A^H )</th>
<th>( A^F )</th>
<th>Entailment</th>
<th>Update ( B/C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_a \land \neg a \land x_b \land \neg b \land x_c \land \neg c )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( A^H_1 = 100101 )</td>
<td>( A^F_1 = a, \neg b, \neg c )</td>
<td>( F \models \neg A^F_1 )</td>
<td>(( x_c ))</td>
</tr>
<tr>
<td>( A^H_2 = 100110 )</td>
<td>( A^F_2 = a, \neg b, c )</td>
<td>( A^F_2 \models F )</td>
<td>(( \neg x_a \lor \neg x_c ))</td>
</tr>
</tbody>
</table>
Example for algorithm 1

\[ H = L \cup B \cup C \]

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]

- SAT oracle query: \( F \land A^F \)

<table>
<thead>
<tr>
<th>( A^H )</th>
<th>( A^F )</th>
<th>Entailment</th>
<th>Update ( B/C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_a \neg a x_b \neg b x_c \neg c )</td>
<td>( A_1^F = a, \neg b, \neg c )</td>
<td>( F \models \neg A_1^F )</td>
<td>( (x_c) )</td>
</tr>
<tr>
<td>( A_2^H = 100110 )</td>
<td>( A_2^F = a, \neg b, c )</td>
<td>( A_2^F \models F )</td>
<td>( (\neg x_a \lor \neg x_c) )</td>
</tr>
<tr>
<td>( A_3^H = 010110 )</td>
<td>( A_3^F = \neg a, \neg b, c )</td>
<td>( F \models \neg A_3^F )</td>
<td>( (x_a \lor x_b) )</td>
</tr>
</tbody>
</table>
Example for algorithm 1

\[ H = L \cup B \cup C \]

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]

- SAT oracle query: \( F \land A^F \)

<table>
<thead>
<tr>
<th>( A^H )</th>
<th>( A^F )</th>
<th>Entailment</th>
<th>Update ( B/C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_a \neg x_a x_b \neg x_b x_c \neg x_c )</td>
<td>( x_c )</td>
<td>( A^F = a, \neg b, \neg c )</td>
<td>( F \not\models \neg A^F )</td>
</tr>
<tr>
<td>( A_1^H = 100101 )</td>
<td>( A_1^F = a, \neg b, \neg c )</td>
<td>( F \models \neg A^F )</td>
<td>( (x_c) )</td>
</tr>
<tr>
<td>( A_2^H = 100110 )</td>
<td>( A_2^F = a, \neg b, c )</td>
<td>( A^F = F )</td>
<td>( (\neg x_a \lor \neg x_c) )</td>
</tr>
<tr>
<td>( A_3^H = 010110 )</td>
<td>( A_3^F = \neg a, \neg b, c )</td>
<td>( F \models \neg A^F )</td>
<td>( (x_a \lor x_b) )</td>
</tr>
<tr>
<td>( A_4^H = 011010 )</td>
<td>( A_4^F = \neg a, b, c )</td>
<td>( A^F = F )</td>
<td>( (\neg x_b \lor \neg x_c) )</td>
</tr>
</tbody>
</table>
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?**

- 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 $F$
  – 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. $M \neg F \models \neg F$, and $\neg M \neg F$ is an implicate of $F$
  – Extract prime implicate
  – Block prime implicate (in $C$)
- Update $H$ and repeat
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 $A^F \models F$, 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 \textbf{minimal} models \( A^H \) of working formula \( H \)
  - Initially \( H = L; C = \emptyset; B = \emptyset \)
  - \textbf{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 \textbf{prime} implicant of \( F \)
  - \textbf{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. \( M^{\neg F} \models \neg F \), and \( \neg M^{\neg F} \) is an \textbf{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 $A^F \models F$, 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. $M^{\neg F} \models \neg F$, 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 $A^F \models F$, 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. $M^{\neg F} \models \neg F$, 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**: $Pl_n(F)$ and prime implicate cover of $F$

$$H \leftarrow \{ (\overline{x}_v \lor \overline{x}_{\overline{v}}) \mid v \in \text{var}(F) \}$$
Algorithm 2

**input**: Formula $F$

**output**: $Pl_n(F)$ and prime implicate cover of $F$

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

**while** true **do**

$$(\text{st}, A^H) \leftarrow \text{MinModel}(H)$$

**if** not st **then** return
Algorithm 2

**input**: Formula $F$

**output**: $Pl_n(F)$ and prime implicate cover of $F$

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

**while true do**

$(\text{st}, A^H) \leftarrow \text{MinModel}(H)$

**if not st then return**

$A^F \leftarrow \text{Map}(A^H)$

$(\text{st}, M^{\neg F}) \leftarrow \text{SAT}(A^F \cup \neg F)$
Algorithm 2

**input**: Formula $F$

**output**: $Pl_n(F)$ and prime implicate cover of $F$

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

**while** true **do**

$(st, A^H) \leftarrow \text{MinModel}(H)$

**if not** $st$ **then return**

$A^F \leftarrow \text{Map}(A^H)$

$(st, M^{\neg F}) \leftarrow \text{SAT}(A^F \cup \neg F)$

**if** $st$ **then**

$\# F \models \neg M^{\neg F}$; i.e. $\neg M^{\neg F}$ is an implicate

$l_e \leftarrow \text{ReduceImplicate}(M^{\neg F}, F)$

$b \leftarrow \{x_l \mid l \in l_e\}$

**else**

$b \leftarrow \{\neg x_l \mid l \in l_e\}$

$H \leftarrow H \cup \{b\}$
Algorithm 2

**input**: Formula $F$

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

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

**while** true **do**

$(st, A^H) \leftarrow \text{MinModel}(H)$

**if** not $st$ **then** return

$A^F \leftarrow \text{Map}(A^H)$

$(st, M^{\neg F}) \leftarrow \text{SAT}(A^F \cup \neg F)$

**if** $st$ **then**

# $F \models \neg M^{\neg F}$; i.e. $\neg M^{\neg F}$ is an implicate

$I_e \leftarrow \text{ReduceImplicate}(M^{\neg F}, F)$

$b \leftarrow \{ x_l \mid l \in I_e \}$

else

# $A^F \models F$; i.e. $A^F$ is a prime implicant

$I_n \leftarrow A^F$

ReportPrimeImplicant($I_n$)

$b \leftarrow \{ \neg x_l \mid l \in I_n \}$

$H \leftarrow H \cup \{ b \}$
Example for algorithm 2

\[ H = L \cup B \cup C \]

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]

- SAT oracle query: \( F \land A^F \)

<table>
<thead>
<tr>
<th>( A^H )</th>
<th>( A^F )</th>
<th>( \neg M \neg F / \neg \text{st} )</th>
<th>( B / C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_a x_{\neg a} x_b x_{\neg b} x_c x_{\neg c} )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example for algorithm 2

\[ H = L \cup B \cup C \]

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]

- SAT oracle query: \( F \land A^F \)

<table>
<thead>
<tr>
<th>( A^H )</th>
<th>( A^F )</th>
<th>( \neg M \neg F / \neg \text{st} )</th>
<th>( B / C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_a x_{\neg a} x_b x_{\neg b} x_c x_{\neg c} )</td>
<td>( A^F_1 = \emptyset )</td>
<td>( \neg a, \neg b, \neg c )</td>
<td>( (x_a \lor x_b) )</td>
</tr>
</tbody>
</table>
Example for algorithm 2

\[ H = L \cup B \cup C \]

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]

- SAT oracle query: \( F \land A^F \)

<table>
<thead>
<tr>
<th>( A^H )</th>
<th>( A^F )</th>
<th>( \neg M^F / \neg \text{st} )</th>
<th>( B / C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_a \land \neg a \land x_b \land \neg b \land x_c \land \neg c )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td>( A_1^F = \emptyset )</td>
<td>( \neg a, \neg b, \neg c )</td>
<td>( (x_a \lor x_b) )</td>
</tr>
<tr>
<td>001000</td>
<td>( A_2^F = b )</td>
<td>( \neg a, b, \neg c )</td>
<td>( (x_c) )</td>
</tr>
</tbody>
</table>
Example for algorithm 2

\[ H = L \cup B \cup C \]

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]

- SAT oracle query: \( F \land A^F \)

<table>
<thead>
<tr>
<th>( A^H )</th>
<th>( A^F )</th>
<th>( \neg M^F / \neg \text{st} )</th>
<th>( B / C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_a \neg a x_b \neg b x_c \neg c )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td>( A_1^F = \emptyset )</td>
<td>( \neg a, \neg b, \neg c )</td>
<td>( (x_a \lor x_b) )</td>
</tr>
<tr>
<td>001000</td>
<td>( A_2^F = b )</td>
<td>( \neg a, b, \neg c )</td>
<td>( (x_c) )</td>
</tr>
<tr>
<td>001010</td>
<td>( A_3^F = b, c )</td>
<td>( \neg \text{st} )</td>
<td>( (\neg x_b \lor \neg x_c) )</td>
</tr>
</tbody>
</table>
Example for algorithm 2

\[ H = L \cup B \cup C \]

\[ F = (((a \land b) \lor (a \land \neg b)) \land c) \lor (b \land c) \]

- SAT oracle query: \( F \land A^F \)

<table>
<thead>
<tr>
<th>( A^H )</th>
<th>( A^F )</th>
<th>( \neg M^F / \neg \text{st} )</th>
<th>( B / C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_a \land \neg a \land x_b \land \neg b \land x_c \land \neg c )</td>
<td>( A_1^F = \emptyset )</td>
<td>( \neg a, \neg b, \neg c )</td>
<td>( (x_a \lor x_b) )</td>
</tr>
<tr>
<td>000000</td>
<td>000000</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>001000</td>
<td>001000</td>
<td>001000</td>
<td>001000</td>
</tr>
<tr>
<td>001010</td>
<td>001010</td>
<td>001010</td>
<td>001010</td>
</tr>
<tr>
<td>001010</td>
<td>001010</td>
<td>001010</td>
<td>001010</td>
</tr>
<tr>
<td>100010</td>
<td>100010</td>
<td>100010</td>
<td>100010</td>
</tr>
</tbody>
</table>
Outline

Background

Related Work

Primes for Non-Clausal Formulae

Results
Experimental setup

- Server: Intel Xeon E5-2630 2.60GHz, 64GByte
- TO: 3600s
- MO: 10 GByte

- Tools:
  - primer: PRIMe compilER
  - zres-tison

- Benchmarks:
  - Quasigroup classification problems: 83
  - Cryptanalysis of the Geffe stream generator: 600
  - Crafted $F_m \lor PHP_n$: 30
    - $F_m = (x_1 \lor y_1) \land \cdots \land (x_m \lor y_m)$
    - $m \in \{10, \ldots, 20\}$
    - $n \in \{6, \ldots, 10\}$
  - Crafted $F_m \lor GT_n$: 30
    - $n \in \{12, \ldots, 20\}$
## Summary of results

<table>
<thead>
<tr>
<th></th>
<th>QG6</th>
<th>Geffe gen.</th>
<th>F+PHP</th>
<th>F+GT</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong># instances</strong></td>
<td>83</td>
<td>600</td>
<td>30</td>
<td>30</td>
<td>743</td>
</tr>
<tr>
<td>ZRes-tison</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>primer-a (PI_n)</td>
<td>53</td>
<td>596</td>
<td>30</td>
<td>26</td>
<td>705</td>
</tr>
<tr>
<td>primer-a (PI_e)</td>
<td>28</td>
<td>588</td>
<td>30</td>
<td>27</td>
<td>673</td>
</tr>
<tr>
<td>primer-b (PI_n)</td>
<td>64</td>
<td>595</td>
<td>30</td>
<td>30</td>
<td>719</td>
</tr>
<tr>
<td>primer-b (PI_e)</td>
<td>30</td>
<td>577</td>
<td>30</td>
<td>27</td>
<td>664</td>
</tr>
</tbody>
</table>
F+PHP scatter plot

![F+PHP scatter plot graph](graph.png)
Comparing algorithms

- primer-b ($P_{In}$)
- primer-a ($P_{In}$)
- primer-a ($P_{Ie}$)
- primer-b ($P_{Ie}$)

CPU time (s) vs. instances

560 580 600 620 640 660 680 700 720

0
500
1000
1500
2000
2500
3000
3500

instances
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?
Conclusions & future work

- Enumeration of prime implicants for non-clausal formulae with SAT oracles
  - Readily applicable to enumeration of prime implicates
- Improvements to proposed algorithms
- Applications of prime enumeration
- Other compilation languages?
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

- Improvements to proposed algorithms

- Applications of prime enumeration

- Other compilation languages?
Conclusions & future work

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