

Computing Compact DNFs Via Reductions in the Egli-Milner Order

Michael Huth and Nir Piterman

The Dutch-British workshop on
Algebra & Coalgebra meet Proof Theory
25-26 January 2010
London, England



Aim

Design algorithm $\Delta(\phi)$ that converts propositional logic formulas ϕ into disjunctive normalform (DNF) such that

- ▶ computation done by structural induction on input
- ▶ term and/or literal size of output DNFs should be minimized
- ▶ minimization relies on single, algebraic rule



Idea

Realize two versions $\Delta_d(\phi)$ and $\Delta_b(\phi)$ of such an algorithm:

- ▶ both convert each sub-formula and its negation into DNFs
- ▶ both use least fixed-point computation to generate DNFs
- ▶ each one has different rule for least fixed-point compactification



Needed algebra

- ▶ Partial order $(\mathbf{3}^n, \preceq)$ is n -th product of $\{0, 1, ?\}$ with $x \preceq y$ iff for all i , $x_i \neq ?$ implies $x_i = y_i$
- ▶ DNF $\neg p_1 \vee (p_1 \wedge \neg p_2) \vee (p_1 \wedge p_3 \wedge p_4)$, e.g., corresponds to $\{0???, 10??, 1?11\} \subseteq \mathbf{3}^4$
- ▶ $\min(X)$ set of \preceq -minimal elements of $X \subseteq \mathbf{3}^n$
- ▶ $(\mathbf{3}^n, \preceq)$ has binary infima $x \sqcap y$ for all $x, y \in \mathbf{3}^n$
- ▶ $x \uparrow y$ iff x and y have upper bound in $(\mathbf{3}^n, \preceq)$; such consistent pairs have supremum $x \sqcup y$



Invariant of algorithm $\Delta(\phi)$

Let (P, N) be the output of $\Delta(\phi)$

Then P and N are DNFs such that

(Inv) “ P is equivalent to ϕ , N is equivalent to $\neg\phi$ ”



Design of algorithm $\Delta(\phi)$

```
 $\Delta(\phi)$  {  
   $\phi$  is  $p_i$ : return  $(\{e_i^1\}, \{e_i^0\})$ ;  
   $\phi$  is  $\neg\psi$ : return swap $(\Delta(\psi))$ ;  
   $\phi$  is  $\psi_1 \wedge \psi_2$ : return conj $(\Delta(\psi_1), \Delta(\psi_2))$ ;  
}
```

```
swap $(X, Y)$ { return  $(Y, X)$ ;}  
conj $((X_1, Y_1), (X_2, Y_2))$  {  
   $P = \min(\{x_1 \sqcup x_2 \mid x_1 \uparrow x_2, x_1 \in X_1, x_2 \in X_2\})$ ;  
   $N = \min(Y_1 \cup Y_2)$ ;  
  return lfp $(P, N)$ ;  
}
```

Crucial ingredient: least fixed-point computation **lfp** (P, N) ; for conjunction



Least fixed-point computation

```
lfp( $P, N$ ) {  
  repeat {  
     $P' = P; N' = N;$   
     $(P, N) = \text{reduce}(P, N);$   
  } until  $((P = P') \& (N = N'))$   
  return  $(P, N);$   
}
```

Intuition: make P and N simpler until they can no longer be simplified



Egli-Milner Order

Given $X, Y \subseteq \mathbf{3}^n$, let

$$X \sqsubseteq Y \text{ iff } (\forall v \in X \exists w \in Y: v \preceq w) \ \& \ (\forall w \in Y \exists v \in X: v \preceq w)$$

Specification of **reduce**: $P \sqsubseteq P'$ and $N \sqsubseteq N'$ after each call to **reduce**(P, N)

So

$$P \sqsubseteq P' \text{ or } N \sqsubseteq N'$$

if least fixed-point computation hasn't yet terminated



“Breadth-first” implementation of **reduce**

```

breadth( $P, N$ ) implements reduce( $P, N$ ) {
  if ( $\exists x \neq y \in P: \forall z \in N: x \sqcap y \not\leq z$ ) {
     $P = P \setminus \{w \in P \mid x \sqcap y \leq w\} \cup \{x \sqcap y\}$ ;
  } elseif ( $\exists x \neq y \in N: \forall z \in P: x \sqcap y \not\leq z$ ) {
     $N = N \setminus \{w \in N \mid x \sqcap y \leq w\} \cup \{x \sqcap y\}$ ;
  }
}

```

Rule: If distinct elements x, y in P have infimum $x \sqcap y$ inconsistent with all terms of N , add $x \sqcap y$ to P , remove all w with $x \sqcap y \prec w$ from P

Symmetric rule for N



“Depth-first” implementation of **reduce**

```

depth( $P, N$ ) implements reduce( $P, N$ ) {
  if ( $\exists x \in P \exists i: x@i$  defined & ( $\forall z \in N: x@i \not\prec z$ )) {
     $P = P \setminus \{w \in P \mid x@i \preceq w\} \cup \{x@i\};$ 
  } elseif ( $\exists x \in N \exists i: x@i$  defined & ( $\forall z \in P: x@i \not\prec z$ )) {
     $N = N \setminus \{w \in N \mid x@i \preceq w\} \cup \{x@i\};$ 
  }
}

```

Definition: For $x \in P$ with $x_i \neq ?$, let $x@i$ be x except that i th coordinate has now value ?

Rule: If $x@i$ defined, $x \in N$, and $x@i$ inconsistent with all terms of N , add $x@i$ to N , remove all w with $x@i \prec w$ from P

Symmetric rule for P



Correctness of $\Delta_b(\phi)$ and $\Delta_d(\phi)$

Show (Inv) by structural induction on ϕ

- ▶ Obvious arguments for atoms p_i and negation $\neg\psi$
- ▶ For conjunction, elementary to show (Inv) for argument pair (P, N) for call **lfp**(P, N) as

```
conj(( $X_1, Y_1$ ), ( $X_2, Y_2$ )) {  
   $P = \min(\{x_1 \sqcup x_2 \mid x_1 \uparrow x_2, x_1 \in X_1, x_2 \in X_2\})$ ;  
   $N = \min(Y_1 \cup Y_2)$ ;  
  return lfp( $P, N$ );  
}
```



Correctness of $\text{lfp}(P, N)$

Suffices to show that output DNFs of $\text{reduce}(P, N)$ equivalent to input DNFs

Key Lemma: If P DNF of some ψ and N DNF of $\neg\psi$, then z is an implicant of P iff z is inconsistent with all elements of N

Apply lemma to each reduction step in $\text{breadth}(P, N)$ or $\text{depth}(P, N)$



DNF minimization not optimal

Let input be DNF with four terms

$$t_1 = 0000 \quad t_2 = 0011 \quad t_3 = 0001 \quad t_4 = 1000$$

Let DNF ϕ be $((t_2 \vee t_3) \vee t_4) \vee t_1$. Then $\Delta_b(\phi)$ has no non-deterministic choice and outputs as P DNF :

$$\{0000, 00?1, 0111\}$$

But $\{000?, 0?11\}$ is equivalent DNF minimal for term size, computed by Δ_b if terms are bracketed differently



Δ_d outputs only prime implicants

$$\Delta_d(\phi) = (P, N)$$

- ▶ Proof by contradiction: $x \in P$ not prime implicant of ϕ .
There is $w \neq x$ with $w \preceq x$ such that w is an implicant of ϕ
- ▶ So there must be i for which $x@i$ is defined and
 $w \preceq x@i \preceq x$
- ▶ But then $x@i$ is implicant of ϕ and so inconsistent with all z
in N by (Inv)
- ▶ Then x would have been removed from P within **lfp[depth]**
in the call $\Delta_d(\phi)$, contradiction



Δ_d outputs DNFs that share all their literals

$$\Delta_d(\phi) = (P, N)$$

- ▶ Proof by contradiction: some p_i occurs in N but not in P
- ▶ Then all elements of P have i th coordinate ?. And there is $x \in N$ such that $x@i$ is defined
- ▶ As x is inconsistent with all $z \in P$, this also holds for $x@i$ as all $z \in P$ have i th coordinate ?
- ▶ Then x would have been removed in $\Delta_d(\phi)$, contradiction



Post-processing Δ_b with $\text{lfp}[\text{depth}]$

Decreases literal size of output and preserves its term size:

- ▶ Claim: any replacement of x by $x@i$, say when $x \in N$, removes only x from N
- ▶ Proof by contradiction: assume $y \in N$ with $x@i \preceq y$ and $x \neq y$
- ▶ As $x@i \preceq x$, this implies $x@i \preceq x \sqcap y$. So $x@i \not\preceq z$ then implies $x \sqcap y \not\preceq z$, for all $z \in P$
- ▶ But then $x \sqcap y$ would have been removed from N in $\Delta_b(\phi)$, contradiction



Δ_b and Δ_d sound and complete for validity

Immediate from (Inv):

Then the respective set P or N is empty



Experimental results

- ▶ $\Delta_d(\phi)$ and $\Delta_b(\phi)$ implemented in C++, no optimizations
- ▶ runs well on randomly generated formulas with up to 100 variables
- ▶ $\Delta_b(\phi)$ computes prime implicants more often than not
- ▶ $\Delta_d(\phi)$ up to six times faster than $\Delta_b(\phi)$ on pigeon-hole benchmark formulas, but orders of magnitude worse than SAT solvers
- ▶ $\Delta(\phi)$ can still minimize DNFs ϕ produced by SAT solver as over-approximation of transition system
- ▶ also used Δ_b to build some DNFs for hitting-set attack for a Chaumian-mix anonymity protocol



Thank You for your kind attention

Questions?

