Recognition of Positive k-Interval Boolean Functions

O. Čepek   D. Kronus   P. Kučera

Charles University in Prague
University of Liège

DIMACS - RUTCOR Workshop on Boolean and Pseudo-Boolean Functions, 2009

Outline

1. Introduction
   - Interval Representations of Boolean Functions
2. Recognition of positive k-interval functions
   - Positive 1-Interval Functions
   - Positive 2-Interval Functions
   - Positive 3-Interval Functions
   - Generalization to Positive k-Interval Functions
3. Conclusion

Integers and Bit Vectors Correspondence

- n-bit vector $\vec{x} \leftrightarrow$ integer $n(\vec{x})$
- significance of bits - x_1 most, x_n least
  $\Rightarrow n(\vec{x}) = \sum_{i=1}^{n} x_i 2^{n-i}$
- let $\pi : \{1, \dots, n\} \to \{1, \dots, n\}$ be a permutation
- then $\vec{x}^\pi$ is a vector of length n such that
  - $x_i^\pi = x_j$, where $\pi(j) = i$

Examples

i	1	2	3
$\pi(i)$	3	2	1

x_1	x_2	x_3	$n(\vec{x})$	$n(\vec{x}^\pi)$
1	1	0	6	3
0	1	1	3	6

Integers and Bit Vectors Correspondence

- n-bit vector $\vec{x} \leftrightarrow$ integer $n(\vec{x})$
- significance of bits - x_1x

((()]
n
n$\vec{x}$)RecogPF55 7.9601

Integers and Bit Vectors Correspondence

- n-bit vector $\vec{x} \leftrightarrow$ integer $n(\vec{x})$
- significance of bits - x_1 most, x_n least
  $\Rightarrow n(\vec{x}) = \sum_{i=1}^{n} x_i 2^{n-i}$
- let $\pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ be a permutation
- then $\vec{x}^\pi$ is a vector of length n such that
  - $x_i^\pi = x_j$, where $\pi(j) = i$

Examples

i	1	2	3
$\pi(i)$	3	2	1

x_1	x_2	x_3	$n(\vec{x})$	$n(\vec{x}^\pi)$
1	1	0	6	3
0	1	1	3	6

Interval Representation of Boolean Functions

Definition

- Boolean function f on n variables is represented by k intervals $[a^1, b^1] < [a^2, b^2] < \ldots < [a^k, b^k]$ of n-bit integers with respect to ordering π of variables if

$$\forall \vec{x} \in \{0,1\}^n : f(\vec{x}) = 1 \Leftrightarrow n(x^\pi) \in \cup_{i=1}^{k} [a^i, b^i]$$

Interval Representation of Boolean Functions

Definition

- Boolean function f on n variables is represented by k intervals $[a^1, b^1] < [a^2, b^2] < \ldots < [a^k, b^k]$ of n-bit integers with respect to ordering π of variables if

$$\forall \vec{x} \in \{0,1\}^n : f(\vec{x}) = 1 \Leftrightarrow n(x^\pi) \in \cup_{i=1}^{k} [a^i, b^i]$$

Interval Representation of Boolean Functions

Definition

- Boolean function f on n variables is represented by k intervals $[a^1, b^1] < [a^2, b^2] < \ldots < [a^k, b^k]$ of n-bit integers with respect to ordering π of variables if

$$\forall \vec{x} \in \{0,1\}^n : f(\vec{x}) = 1 \Leftrightarrow n(x^\pi) \in \cup_{i=1}^{k} [a^i, b^i]$$

Interval Representation of Boolean Functions

Definition

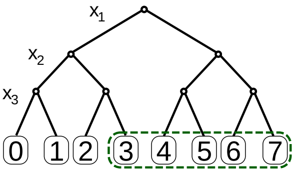
- Boolean function f on n variables is represented by k intervals $[a^1, b^1] < [a^2, b^2] < \ldots < [a^k, b^k]$ of n-bit integers with respect to ordering π of variables if

$$\forall \vec{x} \in \{0,1\}^n : f(\vec{x}) = 1 \Leftrightarrow n(x^\pi) \in \cup_{i=1}^{k} [a^i, b^i]$$

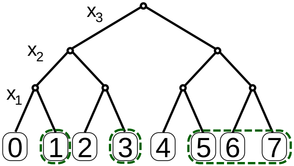
Example (1)

Example

$\mathcal{F} = x_1 \vee x_2 x_3$



ordering $x_1, x_2, x_3 \rightarrow$ interval $[3, 7]$



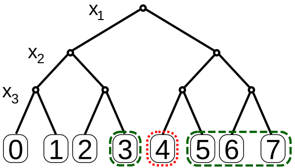
ordering $x_3, x_2, x_1 \rightarrow$ 3 intervals ([1], [3] and $[5, 7]$)

Example (2)

Example

$\mathcal{F} = x_1 x_2 \vee x_2 x_3 \vee x_1 x_3$

Variables are symmetrical $\rightarrow$ all orderings are equivalent.



cannot be represented by 1 interval, only by 2 ([3] and [5, 7,])

Definition

Boolean function f is called k-interval, if it can be represented by at most k intervals (with respect to a suitable ordering).

- Introduced in [Schieber et al., 05] where minimal DNF representations of 1-interval functions were studied.

Definition

Boolean function f is called k-interval, if it can be represented by at most k intervals (with respect to a suitable ordering).

- Introduced in [Schieber et al., 05] where minimal DNF representations of 1-interval functions were studied.

Recognition of positive k-interval functions

Problem:

- input: positive prime DNF $\mathcal{F}$ representing function f, positive integer k
- output: ordering π and intervals $[a_1, b_1] \dots [a_m, b_m]$, $m \leq k$, representing f w.r.t. π or **NO** when f is not k-interval

Recognition of Positive 1-Interval Functions

- if f is 1-interval then there must exist x_i such that one of the following conditions is satisfied:

1) $\mathcal{F}$ contains linear term x_i

2) $\mathcal{F}$ contains x_i in all terms

- the input DNF represents 1-interval function $\Leftrightarrow \mathcal{F}[x_i := 0]$ (resp. $\mathcal{F}[x_i := 1]$) represents 1-interval function

Theorem

Positive 1-interval functions can be recognized in $O(l)$.

Recognition of Positive 1-Interval Functions

- if f is 1-interval then there must exist x_i such that one of the following conditions is satisfied:

1) $\mathcal{F}$ contains linear term x_i

2) $\mathcal{F}$ contains x_i in all terms

- the input DNF represents 1-interval function $\Leftrightarrow \mathcal{F}[x_i := 0]$ (resp. $\mathcal{F}[x_i := 1]$) represents 1-interval function

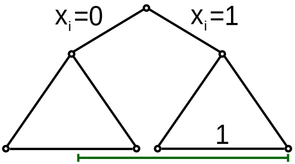
Theorem

Positive 1-interval functions can be recognized in $O(l)$.

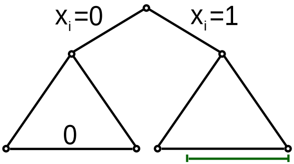
Recognition of Positive 1-Interval Functions

- if f is 1-interval then there must exist x_i such that one of the following conditions is satisfied:

1) $\mathcal{F}$ contains linear term x_i



2) $\mathcal{F}$ contains x_i in all terms



- the input DNF represents 1-interval function $\Leftrightarrow \mathcal{F}[x_i := 0]$ (resp. $\mathcal{F}[x_i := 1]$) represents 1-interval function

Theorem

Positive 1-interval functions can be recognized in $O(l)$.

Recognition of Positive 1-Interval Functions

Recognition of Positive 1-Interval Functions

- if f is 1-interval then there must exist x_i such that one of the following conditions is satisfied:

1) $\mathcal{F}$ contains linear term x_i

2) $\mathcal{F}$ contains x_i in all terms

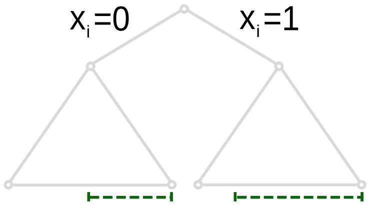
- the input DNF represents 1-interval function $\Leftrightarrow \mathcal{F}[x_i := 0]$ (resp. $\mathcal{F}[x_i := 1]$) represents 1-interval function

Theorem

Positive 1-interval functions can be recognized in $O(l)$.

Recognition of Positive 2-Interval Functions

What happens when none of the conditions 1) and 2) is satisfied?

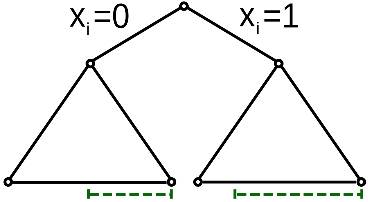


$\Rightarrow \mathcal{F}$ represents 2-interval $\Leftrightarrow \exists$ i: $\mathcal{F}[x_i := 0]$ and $\mathcal{F}[x_i := 1]$ represent 1-interval functions w.r.t. the same ordering π

- How to find such a variable x_i?

Recognition of Positive 2-Interval Functions

What happens when none of the conditions 1) and 2) is satisfied?

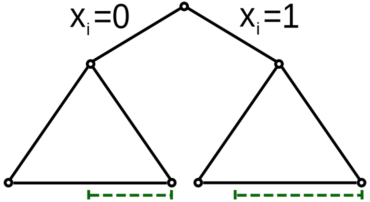


$\Rightarrow \mathcal{F}$ represents 2-interval $\Leftrightarrow \exists$ i: $\mathcal{F}[x_i := 0]$ and $\mathcal{F}[x_i := 1]$ represent 1-interval functions w.r.t. the same ordering π

- How to find such a variable x_i?

Recognition of Positive 2-Interval Functions

What happens when none of the conditions 1) and 2) is satisfied?

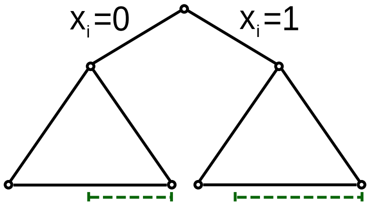


$\Rightarrow \mathcal{F}$ represents 2-interval $\Leftrightarrow \exists$ i: $\mathcal{F}[x_i := 0]$ and $\mathcal{F}[x_i := 1]$ represent 1-interval functions w.r.t. the same ordering π

- How to find such a variable x_i?

Recognition of Positive 2-Interval Functions

What happens when none of the conditions 1) and 2) is satisfied?



$\Rightarrow \mathcal{F}$ represents 2-interval $\Leftrightarrow \exists$ i: $\mathcal{F}[x_i := 0]$ and $\mathcal{F}[x_i := 1]$ represent 1-interval functions w.r.t. the same ordering π

- How to find such a variable x_i?

Recognition of Positive 2-Interval Functions

1. Iterate over all variables and try out.
2. Smarter choice...

Theorem

Let $\mathcal{F}$ be a positive prime DNF representing f which is not 1-interval, moreover none of the conditions 1) or 2) is satisfied in $\mathcal{F}$. Then it suffices to try branching on one of variables x, y for which $\mathcal{F}$ has the following form.

$$\mathcal{F} = xy \vee x\mathcal{G} \vee y\mathcal{H}.$$

Recognition of Positive 2-Interval Functions

1. Iterate over all variables and try out.
2. Smarter choice...

Theorem

Let $\mathcal{F}$ be a positive prime DNF representing f which is not 1-interval, moreover none of the conditions 1) or 2) is satisfied in $\mathcal{F}$. Then it suffices to try branching on one of variables x, y for which $\mathcal{F}$ has the following form.

$$\mathcal{F} = xy \vee x\mathcal{G} \vee y\mathcal{H}.$$

Recognition of Positive 2-Interval Functions

1. Iterate over all variables and try out.
2. Smarter choice...

Theorem

Let $\mathcal{F}$ be a positive prime DNF representing f which is not 1-interval, moreover none of the conditions 1) or 2) is satisfied in $\mathcal{F}$. Then it suffices to try branching on one of variables x, y for which $\mathcal{F}$ has the following form.

$$\mathcal{F} = xy \vee x\mathcal{G} \vee y\mathcal{H}.$$

Recognition of Positive 2-Interval Functions

Algorithm has two phases:

1. same as the algorithm recognizing positive 1-interval functions, i.e. based on conditions 1) and 2)
2. choose candidate variable for branching (it suffices to try one) and perform synchronously the recognition algorithm for positive 1-interval functions on both subtrees.

Theorem

Positive 2-interval functions can be recognized in $O(l)$.

Recognition of Positive 2-Interval Functions

Algorithm has two phases:

1. same as the algorithm recognizing positive 1-interval functions, i.e. based on conditions 1) and 2)
2. choose candidate variable for branching (it suffices to try one) and perform synchronously the recognition algorithm for positive 1-interval functions on both subtrees.

Theorem

Positive 2-interval functions can be recognized in $O(l)$.

Recognition of Positive 2-Interval Functions

Algorithm has two phases:

1. same as the algorithm recognizing positive 1-interval functions, i.e. based on conditions 1) and 2)
2. choose candidate variable for branching (it suffices to try one) and perform synchronously the recognition algorithm for positive 1-interval functions on both subtrees.

Theorem

Positive 2-interval functions can be recognized in $O(l)$.

Recognition of Positive 3-Interval Functions

Phases of algorithm:

1. Based on conditions 1) and 2)...
2. Choose candidate for branching (don't know how...)

3. Synchronously recognize 1-interval and 2-interval functions in subtrees.

Recognition of Positive 3-Interval Functions

Phases of algorithm:

1. Based on conditions 1) and 2)...
2. Choose candidate for branching (don't know how...)

3. Synchronously recognize 1-interval and 2-interval functions in subtrees.

Recognition of Positive 3-Interval Functions

Phases of algorithm:

1. Based on conditions 1) and 2)...
2. Choose candidate for branching (don't know how...)

3. Synchronously recognize 1-interval and 2-interval functions in subtrees.

Recognition of Positive 3-Interval Functions

Implementation:

- For the time being all the candidates for branching have to be tried out
  - First branching.
  - Even in the case of 2-interval function in a subtree because it might actually be 1-interval function but there might be no ordering suitable for both subtrees.

Theorem

Positive 3-interval functions can be recognized in $O(n^2 l)$.

Generalization to Positive k-Interval Functions

- In order to have at most k intervals we can branch only $k-1$ times.
- For the time being we have to try all remaining variables at each point of branching.
- On any level if every subtree satisfies one of the conditions 1) or 2) for the same variable we can proceed without branching using such variable.
- Synchronization of ordering in several subtrees costs alltogether $O(kl)$.

Theorem

Positive k-interval functions can be recognized in $O(kn^{k-1}l)$.

Generalization to Positive k-Interval Functions

- In order to have at most k intervals we can branch only $k-1$ times.
- For the time being we have to try all remaining variables at each point of branching.
- On any level if every subtree satisfies one of the conditions 1) or 2) for the same variable we can proceed without branching using such variable.
- Synchronization of ordering in several subtrees costs alltogether $O(kl)$.

Theorem

Positive k-interval functions can be recognized in $O(kn^{k-1}l)$.

Generalization to Positive k-Interval Functions

- In order to have at most k intervals we can branch only $k-1$ times.
- For the time being we have to try all remaining variables at each point of branching.
- On any level if every subtree satisfies one of the conditions 1) or 2) for the same variable we can proceed without branching using such variable.
- Synchronization of ordering in several subtrees costs alltogether $O(kl)$.

Theorem

Positive k-interval functions can be recognized in $O(kn^{k-1}l)$.

Generalization to Positive k-Interval Functions

- In order to have at most k intervals we can branch only $k-1$ times.
- For the time being we have to try all remaining variables at each point of branching.
- On any level if every subtree satisfies one of the conditions 1) or 2) for the same variable we can proceed without branching using such variable.
- Synchronization of ordering in several subtrees costs alltogether $O(kl)$.

Theorem

Positive k-interval functions can be recognized in $O(kn^{k-1}l)$.

Generalization to Positive k-Interval Functions

- In order to have at most k intervals we can branch only $k-1$ times.
- For the time being we have to try all remaining variables at each point of branching.
- On any level if every subtree satisfies one of the conditions 1) or 2) for the same variable we can proceed without branching using such variable.
- Synchronization of ordering in several subtrees costs alltogether $O(kl)$.

Theorem

Positive k-interval functions can be recognized in $O(kn^{k-1}l)$.

Summary

Open problems:

- Is it possible to eliminate the iteration over all variables at each branching point?
- Is it possible to construct a polynomial (in size of input and output) algorithm recognizing positive k-interval functions?