Interval Representations of Boolean Functions

Doctoral Thesis

David Kronus

Department of Theoretical Computer Science and Mathematical Logic
Faculty of Mathematics and Physics
Charles University in Prague, Czech Republic

Doctoral Thesis defense, 2007
Outline

1. Introduction
   - Interval Representation of Boolean Functions
   - Motivation

2. Studied Problems

3. Conclusions
Basic Definitions

Definition

- **Boolean function** on \( n \) variables is a mapping \( \{0, 1\}^n \mapsto \{0, 1\} \).
- Literal is a negated or non-negated propositional variable.
- Term is a conjunction of literals.
- Disjunctive normal form (DNF) is a disjunction of terms.

- Every Boolean function can be represented by a DNF.
Basic Definitions

Definition

- **Boolean function** on $n$ variables is a mapping \( \{0, 1\}^n \mapsto \{0, 1\} \).
- **Literal** is a negated or non-negated propositional variable.
- **Term** is a conjunction of literals.
- **Disjunctive normal form (DNF)** is a disjunction of terms.

Every Boolean function can be represented by a DNF.
Basic Definitions

Definition

- **Boolean function** on \( n \) variables is a mapping \( \{0, 1\}^n \mapsto \{0, 1\} \).
- **Literal** is a negated or non-negated propositional variable.
- **Term** is a conjunction of literals.
- **Disjunctive normal form (DNF)** is a disjunction of terms.

- Every Boolean function can be represented by a DNF.
Definition

- **Boolean function** on $n$ variables is a mapping 
  $\{0, 1\}^n \mapsto \{0, 1\}$.
- **Literal** is a negated or non-negated propositional variable.
- **Term** is a conjunction of literals.
- **Disjunctive normal form (DNF)** is a disjunction of terms.

Every Boolean function can be represented by a DNF.
Definition

- **Boolean function** on $n$ variables is a mapping $\{0, 1\}^n \mapsto \{0, 1\}$.
- **Literal** is a negated or non-negated propositional variable.
- **Term** is a conjunction of literals.
- **Disjunctive normal form (DNF)** is a disjunction of terms.

Every Boolean function can be represented by a DNF.
Integers and Bit Vectors Correspondence

- $n$-bit vector $\vec{x} \Leftrightarrow$ binary representation of integer $n(\vec{x})$
- Significance of bits:
  - $x_1$ is the most significant bit
  - $x_n$ is the least significant bit.
- $n(\vec{x}) = \sum_{i=1}^{n} x_i 2^{n-i}$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$n(\vec{x})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
Permutations of Bits

- Let \( \pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\} \) be a permutation.
- Let \( \vec{x} \) be an \( n \)-bit vector.
- Then \( \vec{x}^\pi \) is a vector of length \( n \) such that
  - \( x_i^\pi = x_j \), where \( \pi(j) = i \).

### Examples

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \pi(i) )</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( n(\vec{x}) )</th>
<th>( n(\vec{x}^\pi) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>
Let \( \pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\} \) be a permutation.

Let \( \vec{x} \) be an \( n \)-bit vector.

Then \( \vec{x}^\pi \) is a vector of length \( n \) such that

- \( x^\pi_i = x_j \), where \( \pi(j) = i \).

**Examples**

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \pi(i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( n(\vec{x}) )</th>
<th>( n(\vec{x}^\pi) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>
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 $\pi$ of variables determining their significancies, if for every $\vec{x} \in \{0, 1\}^n$ we get $f(\vec{x}) = 1$, if and only if $n(\vec{x}^\pi) \in \bigcup_{i=1}^k [a^i, b^i]$.

- Boolean function $f$ is called $k$-interval, if it can be represented by at most $k$ intervals (with respect to a suitable ordering).
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 $\pi$ of variables determining their significancies, if for every $\vec{x} \in \{0, 1\}^n$ we get $f(\vec{x}) = 1$, if and only if $n(\vec{x}^\pi) \in \bigcup_{i=1}^{k}[a^i, b^i]$.

- Boolean function $f$ is called $k$-interval, if it can be represented by at most $k$ intervals (with respect to a suitable ordering).
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 $\pi$ of variables determining their significances, if for every $\vec{x} \in \{0, 1\}^n$ we get $f(\vec{x}) = 1$, if and only if $n(\vec{x}^\pi) \in \bigcup_{i=1}^{k} [a^i, b^i]$.

- Boolean function $f$ is called $k$-interval, if it can be represented by at most $k$ intervals (with respect to a suitable ordering).
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 $\pi$ of variables determining their significancies, if for every $\vec{x} \in \{0, 1\}^n$ we get $f(\vec{x}) = 1$, if and only if $n(\vec{x}^{\pi}) \in \bigcup_{i=1}^k [a^i, b^i]$.

- Boolean function $f$ is called $k$-interval, if it can be represented by at most $k$ intervals (with respect to a suitable ordering).
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 $\pi$ of variables determining their significancies, if for every $\vec{x} \in \{0, 1\}^n$ we get $f(\vec{x}) = 1$, if and only if $n(\vec{x}^\pi) \in \bigcup_{i=1}^{k} [a^i, b^i]$.

- Boolean function $f$ is called $k$-interval, if it can be represented by at most $k$ intervals (with respect to a suitable ordering).
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 $\pi$ of variables determining their significancies, if for every $\vec{x} \in \{0, 1\}^n$ we get $f(\vec{x}) = 1$, if and only if $n(\vec{x}^\pi) \in \bigcup_{i=1}^k [a^i, b^i]$.

Boolean function $f$ is called $k$-interval, if it can be represented by at most $k$ intervals (with respect to a suitable ordering).
Examples of 1-Interval and 2-Interval Functions

Example

1-Interval function $\mathcal{F} = x_1 \lor x_2x_3$
- considering ordering $x_1, x_2, x_3$
- $\mathcal{F}$ can be represented by $[3, 7]$.

Example

2-interval function $\mathcal{G} = x_1x_2 \lor x_2x_3 \lor x_1x_3$
- all variables of $\mathcal{G}$ are symmetrical
- $\mathcal{G}(0, 1, 1) = \mathcal{G}(1, 0, 1) = 1$
- $\mathcal{G}(1, 0, 0) = 0$
- $\mathcal{G}$ can be represented by $[3], [5, 7]$.
Examples of 1-Interval and 2-Interval Functions

Example

1-Interval function $\mathcal{F} = x_1 \lor x_2 x_3$
- considering ordering $x_1, x_2, x_3$
- $\mathcal{F}$ can be represented by $[3, 7]$.

Example

2-interval function $\mathcal{G} = x_1 x_2 \lor x_2 x_3 \lor x_1 x_3$
- all variables of $\mathcal{G}$ are symmetrical
- $\mathcal{G}(0, 1, 1) = \mathcal{G}(1, 0, 1) = 1$
- $\mathcal{G}(1, 0, 0) = 0$
- $\mathcal{G}$ can be represented by $[3], [5, 7]$.
Motivation

Introduced in [Schieber et al., 05]

- Input data generation for HW and SW testing.

Example (Task)

Generate input data to test a given program on a selected path.

We need to:

1. assemble a predicate expressing the conditions on the input
2. solve this predicate to get the input data values.

Universal predicate representation - DNF

The shorter is the DNF, the faster it is to solve the predicate

⇒ minimal DNF representation of interval conditions.
Motivation

- Introduced in [Schieber et al., 05]
  - Input data generation for HW and SW testing.

Example (Task)

Generate input data to test a given program on a selected path.

- We need to:
  1. assemble a predicate expressing the conditions on the input
  2. solve this predicate to get the input data values.

- Universal predicate representation - DNF
- The shorter is the DNF, the faster it is to solve the predicate
  - $\Rightarrow$ minimal DNF representation of interval conditions.
Motivation

- Introduced in [Schieber et al., 05]
  - Input data generation for HW and SW testing.

Example (Task)

Generate input data to test a given program on a selected path.

We need to:

1. assemble a predicate expressing the conditions on the input
2. solve this predicate to get the input data values.

Universal predicate representation - DNF

The shorter is the DNF, the faster it is to solve the predicate

⇒ minimal DNF representation of interval conditions.
Motivation

- Introduced in [Schieber et al., 05]
  - Input data generation for HW and SW testing.

Example (Task)
Generate input data to test a given program on a selected path.

- We need to:
  1. assemble a predicate expressing the conditions on the input
  2. solve this predicate to get the input data values.
- Universal predicate representation - DNF
- The shorter is the DNF, the faster it is to solve the predicate
  - $\Rightarrow$ minimal DNF representation of interval conditions.
1. Introduction

2. Studied Problems
   - Recognition of Interval Functions
   - Interval Extensions of Partially Defined Boolean Functions
   - Best-Fit Extensions of pdBf

3. Conclusions
Definition

Problem **Recognition**($C$), where $C$ is a class of Boolean functions, is:

- Given DNF $\mathcal{F}$ decide whether function $f$ represented by $\mathcal{F}$ belongs to $C$.

Fundamental property, usually polynomial-time algorithm is required to consider $C$ practically usable.
Definition

Problem **Recognition(C)**, where \( C \) is a class of Boolean functions, is:

- Given DNF \( \mathcal{F} \) decide whether function \( f \) represented by \( \mathcal{F} \) belongs to \( C \).

- Fundamental property, usually polynomial-time algorithm is required to consider \( C \) practically usable.
Motivation for the Recognition problem

Interval representation

1. may be efficient
   - in space - very short
   - in time - efficient evaluation in any vector
   - ⇒ minimization of Boolean functions,

2. may reveal additional characteristics of a function
   - ⇒ logical analysis of data, data mining.
Motivation for the Recognition problem

Interval representation

1. may be efficient
   - in space - very short
   - in time - efficient evaluation in any vector
   - $\Rightarrow$ minimization of Boolean functions,

2. may reveal additional characteristics of a function
   - $\Rightarrow$ logical analysis of data, data mining.
The General Case

Theorem

Recognition of $k$-interval functions is

- **co-NP-hard**
  - when $k$ is a part of input
  - when $k$ is a constant ($k \geq 1$),

- **co-NP-complete when ordering is fixed in advance.**
Theorem

Recognition of 1-interval functions can be solved for

- positive and negative functions in $O(I)$,
- functions satisfying quite strong conditions (including polynomial-time solvable satisfiability) in polynomial time.

- Input DNF must be prime.
Recognition of Renamable 1-Interval Functions

Definition

DNF $\mathcal{F}$ represents renamable 1-interval function if there is a set of variables $S$ such that DNF $\mathcal{F}^S$ (formed from $\mathcal{F}$ by switching positive and negative literals of variables from $S$) represents 1-interval function.

- Recognition can be solved by slight modifications of the algorithms for non-renamable variants.
Recognition of Renamable 1-Interval Functions

Definition

DNF $\mathcal{F}$ represents renamable 1-interval function if there is a set of variables $S$ such that DNF $\mathcal{F}^S$ (formed from $\mathcal{F}$ by switching positive and negative literals of variables from $S$) represents 1-interval function.

- Recognition can be solved by slight modifications of the algorithms for non-renamable variants.
Other results

- Recognition of 2-interval functions can be solved in $O(l)$ for positive and negative functions.
- Recognition of $k$-interval functions w.r.t. a fixed ordering can be solved by polynomially incremental algorithm for positive and negative functions. Each interval is output in $O(l)$.
- Recognition of $k$-interval functions can be solved in $O(kl)$ for 2-monotonic functions.
Partially Defined Boolean Function

Definition

Partially defined Boolean function (pdBf) is pair \((T, F)\)

- \(T, F \subset \{0, 1\}^n\)
- \(\vec{x} \in T \mapsto 1\)
- \(\vec{x} \in F \mapsto 0\)
Definition

Extension of pdBf \((T, F)\) is total Boolean function \(f\) such that

- for every \(\vec{x} \in T\) : \(f(\vec{x}) = 1\)
- for every \(\vec{x} \in F\) : \(f(\vec{x}) = 0\)

- Extension of pdBf \((T, F)\) exists \(\iff T \cap F = \emptyset\)
- We are interested in an extension with some specific properties
  - implied by the origin of data
  - efficiency.
Extension of pdBf

**Definition**

Extension of pdBf \((T, F)\) is total Boolean function \(f\) such that

- for every \(\vec{x} \in T\) : \(f(\vec{x}) = 1\)
- for every \(\vec{x} \in F\) : \(f(\vec{x}) = 0\)

- Extension of pdBf \((T, F)\) exists \(\iff T \cap F = \emptyset\)
- We are interested in an extension with some specific properties
  - implied by the origin of data
  - efficiency.
Extension of pdBf

Definition

Extension of pdBf \((T, F)\) is total Boolean function \(f\) such that

- for every \(\vec{x} \in T\) : \(f(\vec{x}) = 1\)
- for every \(\vec{x} \in F\) : \(f(\vec{x}) = 0\)

- Extension of pdBf \((T, F)\) exists \(\iff T \cap F = \emptyset\)
- We are interested in an extension with some specific properties
  - implied by the origin of data
  - efficiency.
Definition

Problem Extension(C), where C is a class of Boolean functions, is: given pdBf (T, F)
- Decide whether there exists extension from class C.
- In affirmative case output such extension.

- Polynomially solvable for e.g. positive, Horn and regular Boolean functions.
- NP-hard for e.g. renamable Horn and 2-monotonic.
- See [Boros et al., 98] for many more results.
Extension Problem

**Definition**

Problem $\text{Extension}(C)$, where $C$ is a class of Boolean functions, is: given pdBf $(T, F)$

- Decide whether there exists extension from class $C$.
- In affirmative case output such extension.

- Polynomially solvable for e.g. positive, Horn and regular Boolean functions.
- NP-hard for e.g. renamable Horn and 2-monotonic.
- See [Boros et al., 98] for many more results.
Motivation for the Extension problem

Suppose:

- We measure the occurrence of some phenomena under various conditions
  - e.g. occurrence of a disease in relation to blood pressure, body temperature...
- This measurement is very expensive or time consuming
  - e.g. the diagnosis of the disease requires an expensive or rare equipment.
- It is cheap and easy to measure the conditions
  - e.g. blood pressure, body temperature...
Motivation for the Extension problem

Suppose:

- We measure the occurrence of some phenomenon under various conditions
  - e.g. occurrence of a disease in relation to blood pressure, body temperature...
- This measurement is very expensive or time consuming
  - e.g. the diagnosis of the disease requires an expensive or rare equipment.
- It is cheap and easy to measure the conditions
  - e.g. blood pressure, body temperature...
Suppose:

- We measure the occurrence of some phenomena under various conditions
  - e.g. occurrence of a disease in relation to blood pressure, body temperature...
- This measurement is very expensive or time consuming
  - e.g. the diagnosis of the disease requires an expensive or rare equipment.
- It is cheap and easy to measure the conditions
  - e.g. blood pressure, body temperature...
How to Save Resources and Time

- After some data have already been measured, we may use this knowledge:
  - find “explanation” of the data
  - use it to predict the occurrence of phenomena
  - use expensive measurement only for confirmation when needed.

- The “explanation” is a function, which
  - agrees with the experimental data
  - can be used for a prediction on data we have not measured
  - can be Boolean after a binarization of data.
Required properties of the explanation function

1. Implied by the origin of data
   - the higher blood pressure $\rightarrow$ the higher probability of disease occurrence

2. Efficiency
   - fast to evaluate on given data.
Theorem

*The Extension problem for the classes of positive, negative, general and renamable 1-interval functions can be solved in* \(O(n \cdot (n + |T| + |F|))\) *time.*
Usually there are errors in measured data.

- These may cause that there is no extension from the specified class of functions
- \( \implies \) we may try to find in the specified class a function which is the “best” approximation.
Best-Fit Problem

- Generalization of the Extension problem.

Definition

Problem Best-Fit(C), where C is a class of Boolean functions, is: given pdBf \((T, F)\)

- Find a function \(f\) from class \(C\) for which the number of vectors where \(f\) and \((T, F)\) disagree is minimized.
- Vectors in \(T \cup F\) may be weighted \(\Rightarrow\) weighted error size.

- Harder than Extension.
- Polynomially solvable for e.g. positive and regular Boolean functions.
- NP-hard for e.g. Horn, renamable Horn and 2-monotonic.
Best-Fit Problem

- Generalization of the Extension problem.

**Definition**

Problem **Best-Fit**(C), where C is a class of Boolean functions, is: given pdBf \((T, F)\)
- Find a function \(f\) from class \(C\) for which the number of vectors where \(f\) and \((T, F)\) disagree is minimized.
- Vectors in \(T \cup F\) may be weighted \(\Rightarrow\) weighted error size.

- Harder than Extension.
- Polynomially solvable for e.g. positive and regular Boolean functions.
- NP-hard for e.g. Horn, renamable Horn and 2-monotonic.
Theorem

- The uniform Best-Fit problem for the classes of positive and negative 1-interval functions is NP-hard.
- The weighted Best-Fit problem for the class of general 1-interval functions is NP-hard.

Can be solved in linear time when a fixed ordering of variables is given in advance.
1 Introduction

2 Studied Problems

3 Conclusions
   - Summary
   - Future work
   - Bibliography
1. The Recognition problem can be solved for:
   - positive and negative 1-interval and 2-interval functions in polynomial time,
   - general and renamable 1-interval functions satisfying strong conditions in polynomial time,
   - for 2-monotonic $k$-interval functions in $O(kl)$.

2. The Extension problem can be solved in polynomial time for positive, negative, general and renamable 1-interval functions.

3. The Best-Fit problem is NP-hard for positive, negative and general 1-interval functions.
Future work

Natural generalizations:

- Recognition for general 2,3,4...-interval functions, positive 3,4,5...-interval functions,
- Extension for 2,3,4...-interval functions,
- Best-fit for 2,3,4...-interval functions.
B. Schieber, D. Geist and Z. Ayal
*Computing the Minimum DNF Representation of Boolean Functions Defined by Intervals.*

E. Boros, T. Ibaraki and K. Makino
*Error-free and Best-fit Extensions of Partially Defined Boolean Functions*