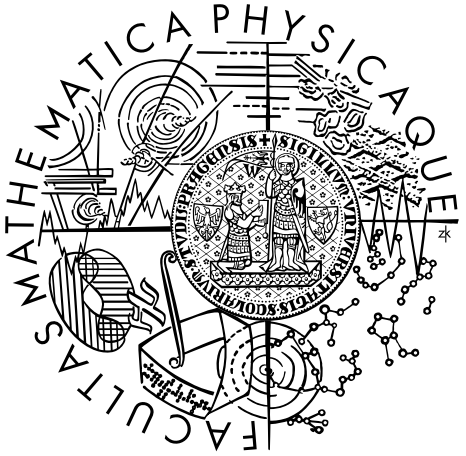
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Filip Bártek

Minimální reprezentace víceintervalových booleovských funkcí

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis: RNDr. Petr Kučera, Ph.D.

Study programme: Informatics

Specialization: Theoretical Computer Science

Prague 2015

I would like to thank my parents for their continuous support throughout my studies, doc. RNDr. Ondřej Čepek, Ph.D. for introducing me to the field of Boolean functions and the questions dealt with in this thesis, my thesis supervisor RNDr. Petr Kučera, Ph.D. for numerous helpful suggestions and a great amount of patience and attention, and my dear friend Jan Balaš for an unexpected helpful discussion.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature

Název práce: Minimální reprezentace víceintervalových booleovských funkcí

Autor: Filip Bártek

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: RNDr. Petr Kučera, Ph.D., Katedra teoretické informatiky a matematické logiky

Abstrakt: Pokud interpretujeme vstupní vektory booleovské funkce jako binárně zapsaná čísla, intervalovou booleovskou funkci $f^n_{[a,b]}$ definujeme tak, že $f^n_{[a,b]}(x) = 1$ právě tehdy když $a \leq x \leq b$. Disjunktivní normální forma je jeden z běžných způsobů reprezentace booleovských funkcí. Minimalizaci DNF reprezentace 1-intervalové booleovské funkce zadané okrajovými hodnotami intervalu lze provést v lineárním čase. Zobecnění na k-intervalové funkce se zdá být složitější. V této diplomové práci diskutujeme komplikace s hledáním optimální DNF reprezentace k-intervalové funkce a představíme $2k$-aproximační algoritmus pro tento problém.

Klíčová slova: booleovská minimalizace, disjunktivní normální forma, intervalové funkce

Title: Minimum representations of Boolean functions defined by multiple intervals

Author: Filip Bártek

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis: RNDr. Petr Kučera, Ph.D., Department of Theoretical Computer Science and Mathematical Logic

Abstract: When we interpret the input vector of a Boolean function as a binary number, we define interval Boolean function $f^n_{[a,b]}$ so that $f^n_{[a,b]}(x) = 1$ if and only if $a \leq x \leq b$. Disjunctive normal form is a common way of representing Boolean functions. Minimization of DNF representation of an interval Boolean function can be performed in linear time. The natural generalization to k-interval functions seems to be significantly harder to tackle. In this thesis, we discuss the difficulties with finding an optimal solution and introduce a $2k$-approximation algorithm.

Keywords: Boolean minimization, disjunctive normal form, interval functions

Contents

Introduction

An n-ary Boolean function takes an n-tuple of Boolean values as input and outputs a single Boolean value. A Boolean function can be represented in various ways. A common way to represent Boolean functions is the truth table, which explicitly lists the output value for each possible input n-tuple [2, Definition 1.2]. For example, the simple Boolean function $f_\wedge^2$ that realizes the binary logical operator of conjunction can be represented by the following table:

x_1	x_2	$f_\wedge^2(x_1, x_2)$
0	0	0
0	1	0
1	0	0
1	1	1

The upper index of $f_\wedge^2$ signifies the arity of the function. In this case it is 2, meaning it is a function of two Boolean variables, or equivalently of pairs of Boolean values, or binary vectors of length 2.

Other representations of Boolean functions include Boolean formulas [2, Definition 1.4], circuits [10, Definition 3.1] and binary decision diagrams [2, Section 1.12.3]. Another possible representation is sets of intervals, which is the one we study in this thesis. We will relate it to the disjunctive normal form (DNF) Boolean formula representation. More specifically, we will look for efficient ways to transform an interval representation to an equivalent DNF representation.

The DNF representation of Boolean functions has been applied in constraint-based software and hardware testing systems [3, 7]. In these systems, DNF formulas act as characteristic functions of constrained variable domains. It can be especially useful to represent the constrained domains with disjoint spanning sets, since these allow fast uniform polling of admissible values [8].

Since the constrained variables are typically numeric and comparison is one of the basic constraints, the variable domains are often constrained to small sets of intervals. Therefore it can be interesting to look for compact DNF representations of intervals.

The notion of interval Boolean functions was introduced by Schieber et al. [8]. A pair of n-bit integers a, b defines the 1-interval function $f_{[a,b]}^n$:

$$f_{[a,b]}^n(x) = 1 \iff a \leq x \leq b$$

Note that such function is the characteristic function of the interval $[a, b]$.

Schieber et al. showed an efficient method to find a minimum disjunctive normal form (DNF) representation of any 1-interval Boolean function given by a pair of endpoints [8].

Since the problem of Boolean minimization is in general Σ_2^p-complete [9], it is an interesting question how difficult it is to minimize DNF representation of a k-interval function, that is a function defined by a set of k intervals. Dubovský investigated the problem in the case of 2-interval functions [4]. Another useful measure is to consider the number of switches of the function, which correspond to the inner interval endpoints (we will give a precise definition in Section 1.9).

In this thesis we review the existing results regarding minimization of DNF representations of single- and multi-interval functions, and then present four new results:

- a simplified algorithm for minimizing DNF representations of 2-switch 2-interval functions (Section 3.1),
- an argument that shows that the technique used for proving the optimality of Schieber et al.'s 1-interval minimization algorithm [8] and Dubovský's 2-switch 2-interval minimization algorithm [4] can not be used in general for functions with 3 or more switches (Section 3.2),
- an simple approximation algorithm that, given the endpoints of intervals of a k-interval function f, finds a DNF representation of f that is at worst $2k$ times larger than an optimal DNF representation (Chapter 4), and
- a more sophisticated approximation algorithm for k-interval functions which performs better than the simple $2k$ approximation algorithm, although we show that the relative improvement diminishes for large k (Chapter 5).

1. Definitions

The basic terminology used in this thesis was introduced by Crama and Hammer [2], Schieber et al. [8], Dubovský [4] and Hušek [5, 6].

1.1 Vector operations

Throughout the thesis we will use vectors over small finite domains extensively. We shall write a vector as a sequence of symbols, for example 00101 is a vector of length 5. The concatenation of two vectors u and v will be denoted simply as uv, for example $00v11$ denotes a vector which is formed as a concatenation of the three vectors 00, v and 11.

Definition 1.1.1 (Component extraction $v^{[i]}$ [8])**.** Let v be a vector of length n and $1 \leq i \leq n$. Then $v^{[i]}$ is the i-th component of v.

Definition 1.1.2 (Subsequence extraction $v^{[a,b]}$ [8])**.** Let v be a vector of length n and $1 \leq a \leq b \leq n$. Then $v^{[a,b]}$ is the subvector of v that starts at a-th position and ends at b-th position ($v^{[a,b]} = v^{[a]}v^{[a+1]} \dots v^{[b-1]}v^{[b]}$).

Notably, $v = v^{[1]} \dots v^{[n]} = v^{[1,n]}$.

Definition 1.1.3 (Symbol repetition $\alpha^{\{n\}}$ [8])**.** Let α be a symbol (for example 0 or 1) and $n \in \{0, 1, \dots\}$. Then $\alpha^{\{n\}}$ is the vector of length n each component of which is equal to α.

For example $0^{\{2\}} = 00$.

1.2 Binary vectors

Definition 1.2.1 (Binary vector [8])**.** We will use the term *binary vector* to denote a vector over the Boolean domain. That is, x is a binary vector if and only if $x \in \{0, 1\}^n$ for some $n \in \{0, 1, \dots\}$. We call such x an *n-bit binary vector*.

Note that an n-bit binary vector x is a binary representation of the natural number $\sum_{i=1}^{n} x^{[i]}2^{n-i} = \sum_{i|x^{[i]}=1} 2^{n-i}$. An n-bit vector corresponds to a number between 0 and $2^n - 1$. We will not differentiate between a number and its binary vector representation. Specifically, we will use the standard linear order of natural numbers to compare binary vectors. Note that the linear order on natural numbers corresponds to lexicographic order on binary vectors:

Observation 1.2.1. *If a and b are n-bit binary vectors, then $a < b$ if and only if there are binary vectors c, a' and b' such that $a = c0a'$ and $b = c1b'$. We call such c the* longest common prefix *of a and b.*

1.3 Boolean functions

Definition 1.3.1 (Boolean function [2])**.** $f : \{0,1\}^n \to \{0,1\}$ is an *n-ary Boolean function.*

Since we will not be dealing with any non-Boolean functions, in the remainder of the text, we will use the terms "function" and "Boolean function" interchangeably.

Definition 1.3.2 (True point [2])**.** Let f be an n-ary Boolean function. An n-bit binary vector x is a *true point* of f if and only if $f(x) = 1$.

Equivalently, we define a *false point* of f as any point x such that $f(x) = 0$.

We'll denote the set of all true points of f as $TP(f)$ and the set of all false points as $FP(f)$.

1.4 DNF representations of Boolean functions

Every n-ary Boolean function f can be expressed as a Boolean formula $\mathcal{F}$ on n variables $x_1, \ldots, x_n$. There's a natural bijection between binary vectors of length n and valuations of n variables (1-bits in the binary vector correspond exactly to 1-valued variables in the valuation).

Definition 1.4.1 (Disjunctive normal form [2])**.** A *literal* is a variable (x_i) or its negation ($\overline{x_i}$). A *term* is an elementary conjunction of literals, that is a conjunction in which every variable appears at most once. A *disjunctive normal form (DNF)* Boolean formula is a disjunction of terms. We will often view terms as sets of literals and DNF formulas as sets of terms.

A DNF formula $\mathcal{F}$ is a *DNF representation* of a Boolean function f if and only if $(\forall v \in \{0,1\}^n)[f(v) = 1 \iff \mathcal{F}(v) \equiv 1]$, where $\mathcal{F}(v)$ is the formula given by substituting each occurrence of a variable x_i in $\mathcal{F}$ with the value $v^{[i]}$ for every $i \in \{1, \ldots, n\}$.

Definition 1.4.2 (Minimum DNF representation [8, 2])**.** A DNF representation of function f is *minimum* if and only if there is no DNF representation of f with fewer terms.

We shall denote the size of a minimum DNF representation of function f by $dnf(f)$.

Note that a function may have more than one minimum DNF representation.

The focus of this thesis is minimizing the number of terms of DNF representations of certain Boolean functions.

1.5 Ternary vectors

Definition 1.5.1 (Ternary vector [8])**.** A *ternary vector* is a vector over the set $\{0, 1, \phi\}$.

There's a natural bijection between ternary vectors and DNF terms (elementary conjunctions of literals). Each of the n components of a ternary vector T corresponds to an occurrence (or its absence) of one of the n variables of a term C:

$T^{[i]}$	$C \cap \{x_i, \overline{x_i}\}$
ϕ	$\emptyset$
1	$\{x_i\}$
0	$\{\overline{x_i}\}$

Note that since C is elementary, it can not contain both x_i and $\overline{x_i}$ for any i.

Namely, a *binary* vector (a ternary vector that does not contain any ϕ) corresponds to a full term (that is one that contains every variable), and the ternary vector $\phi^{\{n\}}$ corresponds to the empty term (tautology).

It is easy to see that a set of ternary vectors, each of length n, corresponds to a DNF *formula* on n variables.

Definition 1.5.2 (Spanning [8])**.** A ternary vector T of length n *spans* a binary vector x of length n if and only if $(\forall i \in \{1, \ldots, n\})[T^{[i]} = \phi \text{ or } T^{[i]} = x^{[i]}]$.

A set of ternary vectors $\mathcal{T}$ *spans* a binary vector x if and only if some vector in $\mathcal{T}$ spans x.

The i-th symbol of a ternary vector constrains the i-th symbol of the spanned binary vector (or, equivalently, the valuation of the i-th variable). If $T^{[i]}$ is a "fixed bit" (that is $T^{[i]} \in \{0, 1\}$), the spanned binary vector x must have the same value in its i-th position, that is $x^{[i]} = T^{[i]}$. If $T^{[i]}$ is the "don't care symbol" ϕ, the spanned binary vector may have any value in its i-th position.

In the correspondence between ternary vectors and terms, spanning corresponds to satisfying. The ternary vector T of length n spans x if and only if $C(x) \equiv 1$, where C is the term on n variables that corresponds to T. Similarly, it is easy to see that a set of ternary vectors $\mathcal{T}$ spans x if and only if $\mathcal{F}(x) \equiv 1$, where $\mathcal{F}$ is the DNF formula on n variables that corresponds to $\mathcal{T}$.

Definition 1.5.3 (Spanned set)**.** Let T be a ternary vector of length n. We denote the set of points spanned by T as $span(T)$:

$$span(T) = \{x \in \{0, 1\}^n | T \text{ spans } x\}$$

Note that a ternary vector with m ϕ-positions spans 2^m binary vectors.

We'll also use a natural extension of spanning to sets of ternary vectors – if $\mathcal{T}$ is a set of ternary vectors, then

$$span(\mathcal{T}) = \bigcup_{T \in \mathcal{T}} span(T)$$

Definition 1.5.4 (Exact spanning)**.** A ternary vector T of length n *spans exactly* a set of n-bit binary vectors S if and only if $span(T) = S$.

T *spans exactly* an n-ary Boolean function f if and only if $span(T) = TP(f)$.

Again, we will generalize *exact spanning* to sets of ternary vectors – the set of ternary vectors $\mathcal{T}$ *spans exactly* S if and only if $span(\mathcal{T}) = S$.

Definition 1.5.5 (Spanning set [8])**.** Let f be an n-ary Boolean function. The set $\mathcal{T}$ of n-bit ternary vectors is a *spanning set* of f if and only if it $\mathcal{T}$ spans exactly f, that is $span(\mathcal{T}) = TP(f)$.

In other words, a spanning set of a function spans all of its true points and none of its false points.

Note that spanning sets of a function correspond to DNF representations of the function. The correspondence preserves size (number of ternary vectors and terms, respectively), so a minimum spanning set corresponds to a minimum DNF representation.

Definition 1.5.6 (Disjoint spanning set [8, Section 4])**.** A spanning set $\mathcal{T}$ is *disjoint* if and only if the spanned sets of its vectors do not overlap, that is

$$(\forall T_i, T_j \in \mathcal{T})[T_i = T_j \text{ or } span(T_i) \cap span(T_j) = \emptyset]$$

In terms of DNF, disjoint spanning sets correspond to DNF formulas such that every valuation satisfies at most one term.

We have introduced two equivalent representations of Boolean functions – one based on ternary vectors and the other being the DNF representation. Table 1.1 shows the corresponding terms side by side.

ternary vector	term
ternary vector set	DNF formula
binary vector	variable valuation
ternary vector *spans* a binary vector	valuation *satisfies* a term
ternary vector set *spans* a binary vector	valuation *satisfies* a DNF formula

Table 1.1: Corresponding terms used in Boolean function representations

Definition 1.5.7 (Complement [8])**.** The complement of a binary symbol α ($\alpha \in \{0, 1\}$), denoted $\overline{\alpha}$, is $1 - \alpha$.

We get the complement of a binary vector x by flipping all of its bits, that is $\overline{x} = (\overline{x_1}, \ldots, \overline{x_n}) = 1^{\{n\}} - x$.

The complement of the ϕ symbol is ϕ ($\overline{\phi} = \phi$).

It follows that we obtain the complement of a ternary vector by flipping all of its fixed bits.

We generalize the notion of complement to sets of ternary vectors – if $\mathcal{T}$ is a set of ternary vectors, then $\overline{\mathcal{T}} = \{\overline{T} | T \in \mathcal{T}\}$.

1.6 Orthogonal sets and coverable functions

Orthogonal set is a useful tool for showing a lower bound of the size of all spanning sets of a function. Orthogonal sets have been used both by Schieber et al. [8] (implicitly) and Dubovský [4] (explicitly) in the proofs of optimality of spanning algorithms. The concept of orthogonal sets was also studied in Čepek et al. [11], albeit using a different notation (see Remark 1.1 below).

Definition 1.6.1 (Orthogonality [4, p. 6])**.** Let x and y be true points of a Boolean function f (that is $f(x) = f(y) = 1$). The vectors x and y are *orthogonal* with respect to f if and only if every ternary vector that spans both x and y spans some false point of f.

A set of true points S is *orthogonal* if and only if all the vectors in S are pairwise orthogonal.[1]

We shall denote the size of a maximum orthogonal set of function f with $ortho(f)$.

The following proposition appears in Dubovský [4]. It was also shown for falsepoint essential sets in conjunctive normal form (CNF) terminology in Čepek et al. [11]. To have the thesis self-contained, we include the proof here as well.

Theorem 1.6.1 ($ortho(f) \leq dnf(f)$ [4, Observation 1.1] [11, Theorem 2.8, Corollary 3.2])**.** *For any Boolean function, the size of its maximum orthogonal set is at most as great as the size of its minimum DNF representation.*

Proof. We will prove the statement by contradiction. Let f be a Boolean function such that $ortho(f) > dnf(f)$. Let V be an orthogonal set of size $ortho(f)$ and $\mathcal{T}$ be any spanning set of size $dnf(f)$ (such spanning set exists because of the correspondence between DNF representations and spanning sets). Since $|\mathcal{T}| < |V|$ and $\mathcal{T}$ spans $TP(f) \supseteq S$, there must be a ternary vector $T \in \mathcal{T}$ that spans two distinct vectors $x, y \in V$. However, since V is orthogonal, x is orthogonal to y, so T necessarily spans a false point, which contradicts the premise that $\mathcal{T}$ is a feasible spanning set of f. □

Remark 1.1. Orthogonal sets were studied in Čepek et al. [11]. Čepek et al., however, use the notion of a falsepoint essential set for the case of conjunctive normal form which in a DNF setting translates into a truepoint essential set. A truepoint essential set $\mathcal{E}(x)$ associated with a true point x of a given function f is a set of implicants of f which are satisfied on x. The fact that the two true points x and y are orthogonal then naturally corresponds to the fact that their associated truepoint essential sets $\mathcal{E}(x)$ and $\mathcal{E}(y)$ are disjoint. An orthogonal set then corresponds to a set of pairwise disjoint truepoint essential sets. The truepoint essential set defined in this way is in fact just a special case of a more general notion of essential set which was introduced in Boros et al. [1]. The results in Čepek et al. [11] show that for our purpose it is enought to consider the truepoint essential sets.

We conclude that the size of any orthogonal set is a lower bound of the size of any (especially minimum) DNF representation. Specifically, if we show a DNF representation of a function and an orthogonal set of the same size, we conclude that the representation is minimum. The orthogonal set certifies the optimality of the DNF representation. This is captured by the notion of coverability defined in Čepek et al. [11]:

Definition 1.6.2 (Coverability [11, Definition 2.9])**.** Let f be a Boolean function. The function f is *coverable* if and only if $ortho(f) = dnf(f)$.

[1] This notion of orthogonality is not related to orthogonal DNF formulas [2, Definition 1.13].

Informally speaking, a coverable function is one for which we have a simple certificate of the optimality of a spanning set. We will see in Chapter 2 and Section 3.1 that all the 1-interval and 2-switch 2-interval functions are coverable. We will show examples of functions which are not coverable in Section 3.2.

Definition 1.6.3 (Separating vector [11, Definition 3.3])**.** Let x, y and z be n-bit binary vectors. We say that z *separates* x and y if and only if for every $i \in \{1, \ldots, n\}$ we have $z^{[i]} = x^{[i]}$ or $z^{[i]} = y^{[i]}$.

The following proposition appears in Čepek et al. [11, Lemma 3.7]. We include the proof for completeness.

Theorem 1.6.2 (Separating false point and orthogonality)**.** *Let x and y be true points of function f. If any false point of f separates x and y, then x and y are orthogonal.*

Proof. Let x and y be true points of an n-ary function f. Let z be a false point of f that separates x and y. Let an arbitrary ternary vector T span x and y. We claim that T necessarily also spans z.

For every position $i \in \{1, \ldots, n\}$ such that $T^{[i]} \neq \phi$ we have $T^{[i]} = x^{[i]} = y^{[i]}$ since T spans both x and y (see Definition 1.5.2). Because z separates x and y, we have $z^{[i]} = x^{[i]} = y^{[i]} = T^{[i]}$.

We conclude that T spans z. □

Note that the inverse implication holds as well [11, Lemma 3.7], but we will not use it in this thesis.

1.7 k-interval Boolean functions

The notation for 1- and 2-interval functions was introduced by Schieber et al. and Dubovský respectively. We shall generalize it to k-interval functions for every $k \geq 0$.

Definition 1.7.1 (k-interval Boolean function)**.** Let $n \geq 0$ and $k \geq 0$. Let $a_1, b_1, \ldots, a_k, b_k$ be n-bit binary vectors such that $0^{\{n\}} \leq a_1 \leq b_1 < a_2 \leq \cdots < a_i \leq b_i < a_{i+1} \leq \cdots < a_k \leq b_k \leq 1^{\{n\}}$. Then $f^n_{[a_1,b_1],\ldots,[a_k,b_k]} : \{0,1\}^n \to \{0,1\}$ is a function defined as follows:

$$f^n_{[a_1,b_1],\ldots,[a_k,b_k]}(x) = \begin{cases} 1 & \text{if } a_i \leq x \leq b_i \text{ for some } i \\ 0 & \text{otherwise} \end{cases}$$

We call $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$ a *k-interval Boolean function* and the vectors a_1, b_1, …, a_k, b_k its *endpoints*.

Note in particular that in case $n = 0$, the 1-interval 0-ary function $f^0_{[\varepsilon,\varepsilon]}$, where ε denotes the binary vector of length 0, is properly defined and equivalent to tautology. Similarly note that for $k = 0$ a 0-interval function $f^n_\emptyset$ is equivalent to contradiction.

Note that the inequalities ensure that the intervals $[a_i, b_i]$ for all indices i are non-empty and do not intersect. The inequalities, however, allow the intervals to be adjoint. This means that a k-interval function may be equivalent to a k'-interval function for some $k' < k$. We will often find it useful to use a less ambiguous notion of the number of intervals of a function:

Definition 1.7.2 (Proper k-interval Boolean function)**.** Let $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$ be a k-interval Boolean function. If the endpoints satisfy the inequalities $b_1 < a_2 - 1$, $\ldots$, $b_i < a_{i+1} - 1$, $\ldots$, $b_{k-1} < a_k - 1$ (in other words, there is at least one false point between any pair of adjacent intervals), we call $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$ a *proper k-interval Boolean function.*

Especially, $f^n_{[a,b]}$ is the function whose true points form the interval $[a, b]$. Table 1.2 shows examples of interval functions.

Interval function	Boolean formula	Description
$f^n_{[1^{\{n\}},1^{\{n\}}]}$	$\bigwedge_i x_i$	Conjunction
$f^n_{[0^{\{n-1\}}1,1^{\{n\}}]}$	$\bigvee_i x_i$	Disjunction
$f^n_{[0^{\{n\}},1^{\{n\}}]}$	1 (an empty term)	Tautology
$f^n_{[0^{\{n-1\}}1,1^{\{n-1\}}0]}$	$\overline{x_1 \ldots x_n} \wedge \overline{\overline{x_1} \ldots \overline{x_n}} \equiv \bigvee_{i \neq j} x_i \overline{x_j}$	Non-equivalence
$f^n_{[0^{\{n\}},0^{\{n\}}],[1^{\{n\}},1^{\{n\}}]}$	$x_1 \ldots x_n \vee \overline{x_1} \ldots \overline{x_n}$	Equivalence

Table 1.2: Examples of interval functions

Definition 1.7.3 ($INT(k)$)**.** For any $k \geq 0$, $INT(k)$ is the class of all proper k-interval Boolean functions.

A k-interval function is typically specified by the $2k$ endpoints of its intervals.

1.8 Minimization of k-interval functions

Having defined all the necessary terms, let us present the central problem we will discuss throughout the thesis:

Problem 1.1 (Minimization of proper k-interval functions)**.** For a fixed $k \geq 0$, the problem *MIN-INT*(k) is defined as follows: Given the endpoints of a proper k-interval function f, find a minimum spanning set of f.

Note that since the problem of general Boolean minimization is Σ_2-hard [9] and every Boolean function is k-interval for some k, we do not expect to be able to find a general polynomial minimization algorithm, that is one that is polynomial with respect to both k (number of intervals) and n (function arity). However, an algorithm polynomial with respect to n if we consider k a constant still may exist and it is an interesting open question whether such algorithm exists.

We will also touch upon the alternative version of the minimization problem that involves disjoint representations:

Problem 1.2 (Disjoint minimization of proper k-interval functions)**.** Given the endpoints of a proper k-interval function f, find a minimum disjoint spanning set of f.

1.9 l-switch Boolean functions

Given a k-interval function, apparently the most important values are the endpoints of the intervals. The important property of an interval endpoint is that it is a place where the value of the function changes (considering proper k-interval functions). We can view these values as places of switch, where f switches its value from 0 to 1 or back.

The notion of switch was introduced by Hušek[2] and it turns out it is very useful when studying 2-interval functions.

Definition 1.9.1 (l-switch function [6, pp. 13-14])**.** Let $l \geq 0$. A Boolean function f is *l-switch* if and only if there are exactly l input vectors x such that $f(x) \neq f(x+1)$.

We call such x a *switch* of f.

Example 1.1 (Switches and intervals). Depending on $f(0)$, an l-switch Boolean function f is proper $\lceil \frac{l}{2} \rceil$-interval in case $f(0) = 0$, or proper $\lceil \frac{l+1}{2} \rceil$-interval in case $f(0) = 1$.

Equivalently, a proper k-interval n-ary function is:

- $(2k)$-switch in case $f(0^{\{n\}}) = f(1^{\{n\}}) = 0$
- $(2k-1)$-switch in case $f(0^{\{n\}}) \neq f(1^{\{n\}})$
- $(2k-2)$-switch in case $f(0^{\{n\}}) = f(1^{\{n\}}) = 1$

[2] Hušek first used the Czech term "zlom" in his thesis [5, p. 13] and later translated the term as "switch" in personal communication [6].

2. 1-interval functions

In this chapter, we will show an efficient way to compute a minimum spanning set of any 1-interval Boolean function defined by the interval endpoints, that is to optimally solve the problem *MIN-INT*(1) (see Problem 1.1). The algorithm was originally shown by Schieber et al. [8] and runs in polynomial time. In the chapters that follow, we will use this algorithm as a procedure in order to span multi-interval functions.

In the remainder of this chapter, a and b ($a \leq b$) will denote the endpoints of the interval in question and n the number of input bits. We will be looking for a minimum spanning set of the function $f^n_{[a,b]}$.

Input n-bit numbers a, b such that $a \leq b$ for $n \geq 0$.

Output A spanning set of $f^n_{[a,b]}$.

The algorithm differentiates various cases of input values. Solving the more difficult cases involves recursive calls, while the simpler cases are solved iteratively.

2.1 Trivial cases

We will first deal with the trivial intervals.

2.1.1 $n = 0$

Since the algorithm is recursive, we need to deal with the degenerate case of 0-bit endpoints. We span such interval using a single ternary vector of length 0. Note that such vector corresponds to the empty term, that is tautology.

From now on, let $n \geq 1$.

2.1.2 $a^{[1]} = b^{[1]}$

If a and b share the leading bit, let c be the longest common prefix of a and b and let j be its length. Note that $j \geq 1$. Let $a' = a^{[j+1,n]}$ and $b' = b^{[j+1,n]}$ ($a = ca'$ and $b = cb'$). Let $\mathcal{T}'$ be the spanning set of the function $f^{n-j}_{[a',b']}$. Since $n - j < n$, we can get such set by recursion. We get the spanning set of $f^n_{[a,b]}$ by prepending c to each of the vectors in $\mathcal{T}'$ ($\mathcal{T} = c\mathcal{T}' = \{cT' | T' \in \mathcal{T}'\}$). It is easy to see that if $\mathcal{T}'$ spans exactly $[a', b']$, then $c\mathcal{T}'$ spans exactly $[ca', cb'] = [a, b]$.

Note that in the special case $a = b$ (that is $c = a = b$ and $j = n$), $\mathcal{T}'$ consists of the single vector of length 0 and $\mathcal{T}$ correctly consists of the single vector $c = a = b$.

From now on let $a^{[1]} \neq b^{[1]}$. Since $a \leq b$, necessarily $a^{[1]} = 0$ and $b^{[1]} = 1$. From now on we may especially assume that $a < b$.

2.1.3 $a = 0^{\{n\}}$ and $b = 1^{\{n\}}$

We span the full interval $[0^{\{n\}}, 1^{\{n\}}]$ with the single ternary vector $\phi^{\{n\}}$, which corresponds to an empty term (tautology).

From now on, let $a > 0^{\{n\}}$ or $b < 1^{\{n\}}$.

2.2 Prefix and suffix case

Since we have dealt with the trivial cases, we are now left with the situation $0^{\{n\}} \leq a < b \leq 1^{\{n\}}$ and $a > 0^{\{n\}}$ or $b < 1^{\{n\}}$.

In this section we shall consider the prefix case, that is $[0^{\{n\}}, b]$ $(a = 0^{\{n\}})$, and the suffix case, that is $[a, 1^{\{n\}}]$ $(b = 1^{\{n\}})$.

2.2.1 Suffix case

The prefix and suffix cases are complementary. There is a size-preserving one-to-one correspondence between the spanning sets of suffix intervals and the spanning sets of prefix intervals. The correspondence complements each of the ternary vectors in the spanning set – the spanning set $\mathcal{T}$ of a suffix interval $[a, 1^{\{n\}}]$ corresponds to the spanning set $\overline{\mathcal{T}} = \{\overline{T} | T \in \mathcal{T}\}$ of the prefix interval $[0^{\{n\}}, \overline{a}]$.

We will prove the correctness of this transformation by a series of simple lemmas.

Lemma 2.2.1 (Complementing flips inequality)**.** *Let a and b be binary vectors of equal length. Then $a \leq b$ if and only if $\overline{b} \leq \overline{a}$.*

Proof. The equivalence follows from Observation 1.2.1.

$$\begin{aligned} a \leq b &\iff a = b \text{ or } (\exists c, a', b')[a = c0a' \text{ and } b = c1b'] \\ &\iff \overline{a} = \overline{b} \text{ or } (\exists \overline{c}, \overline{a'}, \overline{b'})[\overline{a} = \overline{c}1\overline{a'} \text{ and } \overline{b} = \overline{c}0\overline{b'}] \\ &\iff \overline{a} \geq \overline{b} \end{aligned}$$

□

Lemma 2.2.2 (Complementing preserves interval membership)**.** *Let x, a and b be binary vectors of equal length. Then $x \in [a, b]$ if and only if $\overline{x} \in [\overline{b}, \overline{a}]$.*

Proof. The equivalence follows from Lemma 2.2.1:

$$\begin{aligned} x \in [a, b] &\iff x \geq a \text{ and } x \leq b \\ &\iff \overline{x} \leq \overline{a} \text{ and } \overline{x} \geq \overline{b} \\ &\iff \overline{x} \in [\overline{b}, \overline{a}] \end{aligned}$$

□

Lemma 2.2.3 (Complementing preserves interval spanning)**.** *Let a and b be n-bit binary vectors and let $\mathcal{T}$ be a set of ternary vectors of length n. Then $\mathcal{T}$ spans $[a, b]$ if and only if $\overline{\mathcal{T}}$ spans $[\overline{b}, \overline{a}]$.*

Proof. The equivalence follows from Lemma 2.2.2:

$$\begin{aligned} \mathcal{T} \text{ spans } [a, b] &\iff (\forall x)[\mathcal{T} \text{ spans } x \iff x \in [a, b]] \\ &\iff (\forall \overline{x})[\overline{\mathcal{T}} \text{ spans } \overline{x} \iff \overline{x} \in [\overline{b}, \overline{a}]] \\ &\iff \overline{\mathcal{T}} \text{ spans } [\overline{b}, \overline{a}] \end{aligned}$$

□

As a special application of Lemma 2.2.3 we get:

$$\mathcal{T} \text{ spans } [a, 1^{\{n\}}] \iff \overline{\mathcal{T}} \text{ spans } [0^{\{n\}}, \overline{a}]$$

Since the correspondence preserves the size of the spanning set, it also preserves optimality. Thus we can use the correspondence to optimally span suffix intervals using a procedure that spans prefix intervals. We will show such procedure in the following section.

2.2.2 Prefix case

Let us proceed to span a prefix interval. Let $a = 0^{\{n\}}$ and $b < 1^{\{n\}}$.

Let c be the n-bit number $b + 1$. Since $b < 1^{\{n\}}$, we do not need more than n bits to encode c.

The algorithm produces one ternary vector for each bit that is set to 1 in c. If o is a position of a 1 in c ($c^{[o]} = 1$), then the corresponding ternary vector is $c^{[1,o-1]}0\phi^{\{n-o\}}$. Thus we get the following spanning set:

$$\mathcal{T} = \{c^{[1,o-1]}0\phi^{\{n-o\}} | c^{[o]} = 1\}$$

Theorem 2.2.1 (Feasibility)**.** *$\mathcal{T}$ spans exactly the interval $[0^{\{n\}}, b]$.*

Proof. To see that every number spanned by $\mathcal{T}$ is in $[0^{\{n\}}, b]$, note that given an index o of a bit which is set to 1 in c, the biggest number spanned by $c^{[1,o-1]}0\phi^{\{n-o\}}$ is $c^{[1,o-1]}01^{\{n-o\}}$ which is still strictly smaller than c.

On the other hand, consider a number x smaller than c and let o be the most significant bit in which x and c differ. It follows that $x^{[1,o-1]} = c^{[1,o-1]}$ and $0 = x^{[o]} < c^{[o]} = 1$. Then $c^{[1,o-1]}0\phi^{\{n-o\}} \in \mathcal{T}$ spans x. □

Theorem 2.2.2 (Optimality)**.** *$\mathcal{T}$ is the minimum spanning set of $[0^{\{n\}}, b]$.*

Proof. We will construct an orthogonal set V of size $|\mathcal{T}|$. With the aid of Theorem 1.6.1, this proves that $\mathcal{T}$ is a minimum spanning set.

Similarly to the spanning vectors, the orthogonal true points correspond to 1-bits of c:

$$V = \{c^{[1,o-1]}0c^{[o+1,n]} | c^{[o]} = 1\}$$

Clearly $|V| = |\mathcal{T}|$. Also note that all points in V are smaller than c, so they are true points.

We need to prove that V is orthogonal. Let $x, y \in V$, $x \neq y$. Let o_x and o_y be the positions of the symbol 1 in c that were used to construct x and y ($x = c^{[1,o_x-1]}0c^{[o_x+1,n]}$, $y = c^{[1,o_y-1]}0c^{[o_y+1,n]}$). Since $x \neq y$, necessarily $o_x \neq o_y$. Since every component i of c clearly matches the respective component in at least one of x, y, the vector c separates x and y by Definition 1.6.3. Having found a separating false point for x and y we conclude that x and y are orthogonal by Theorem 1.6.2.

Table 2.1 shows a more detailed comparison of the corresponding subvectors of x, y and c in case $o_x < o_y$.

We conclude that V is orthogonal and since $|\mathcal{T}| = |V|$, $\mathcal{T}$ is a minimum spanning set by Theorem 1.6.1. □

	$[1, o_x - 1]$	$[o_x]$	$[o_x + 1, o_y - 1]$	$[o_y]$	$[o_y + 1, n]$
x	$c^{\underdot{[1,o_x-1]}}$	0	$c^{\underdot{[o_x+1,o_y-1]}}$	$c^{\underdot{[o_y]}}$	$c^{\underdot{[o_y+1,n]}}$
y	$c^{\underdot{[1,o_x-1]}}$	$c^{\underdot{[o_x]}}$	$c^{\underdot{[o_x+1,o_y-1]}}$	0	$c^{\underdot{[o_y+1,n]}}$
c	$c^{[1,o_x-1]}$	$c^{[o_x]}$	$c^{[o_x+1,o_y-1]}$	$c^{[o_y]}$	$c^{[o_y+1,n]}$

Table 2.1: Subvectors of x, y and c in case $o_x < o_y$. The $\underdot{\text{emphasized}}$ subvectors of x and y match their counterparts in c.

Corollary 2.2.1. *Every prefix and every suffix function is coverable.*

Corollary 2.2.2. *For every prefix function $f^n_{[0^{\{n\}},b]}$, there is an orthogonal set V of size $dnf(f^n_{[0^{\{n\}},b]})$ such that every pair of vectors $x, y \in V$, $x \neq y$, is separated the false point $b + 1$ (see Definition 1.6.3).*

It is easy to see that the spanning set $\mathcal{T}$ produced by the algorithm is disjoint. Since it is minimum in general, it is a minimum disjoint spanning set too.

2.3 General case

Having solved the trivial and prefix and suffix cases, we are left with the situation $0^{\{n\}} < a < b < 1^{\{n\}}$.

Since we have handled the case $a^{[1]} = b^{[1]}$ in Section 2.1.2, we may assume that $a^{[1]} = 0$ and $b^{[1]} = 1$. This restriction leaves us with four possible combinations of pairs of leading bits of a and b:

1. $a^{[1,2]} = 01$, $b^{[1,2]} = 10$
2. $a^{[1,2]} = 00$, $b^{[1,2]} = 10$
3. $a^{[1,2]} = 01$, $b^{[1,2]} = 11$
4. $a^{[1,2]} = 00$, $b^{[1,2]} = 11$

Following Schieber et al. [8], we will deal with each of these cases separately. Schieber et al.'s proofs of feasibility and optimality of the following algorithm are rather technical and out of scope of this text. Note, however, that the proof of optimality is implicitly based on building an orthogonal set of the same size as the computed spanning set.

In the sections that follow, we will denote $a^{[3,n]}$ with $\hat{a}$ ($a = a^{[1,2]}\hat{a}$) and $b^{[3,n]}$ with $\hat{b}$ ($b = b^{[1,2]}\hat{b}$).

2.3.1 $a^{[1,2]} = 01$ and $b^{[1,2]} = 10$

In this case we span the two sub-intervals $[01\hat{a}, 011^{\{n-2\}}]$ and $[100^{\{n-2\}}, 10\hat{b}]$ separately.

The endpoints of the sub-interval $[01\hat{a}, 011^{\{n-2\}}]$ share the leading bit 0. This means that this sub-interval is immediately reduced to the suffix instance $[1\hat{a}, 11^{\{n-2\}}]$, which can be spanned using the suffix algorithm show in Section 2.2.

The remaining sub-interval $[100^{\{n-2\}}, 10\hat{b}]$ is reduced to $[00^{\{n-2\}}, 0\hat{b}]$, which is a prefix interval.

2.3.2 $a^{[1,2]} = 00$ and $b^{[1,2]} = 10$

In this case, we divide the interval into three sub-intervals:

- $[00\hat{a}, 001^{\{n-2\}}]$
- $[010^{\{n-2\}}, 011^{\{n-2\}}]$
- $[100^{\{n-2\}}, 10\hat{b}]$

The sub-interval $[010^{\{n-2\}}, 011^{\{n-2\}}]$ is spanned by the single ternary vector $01\phi^{\{n-2\}}$.

The sub-intervals $[00\hat{a}, 001^{\{n-2\}}]$ and $[100^{\{n-2\}}, 10\hat{b}]$ are spanned together as follows:

1. Recursively solve the $(n-1)$-bit instance $[0\hat{a}, 1\hat{b}] = [a^{[1]}a^{[3,n]}, b^{[1]}b^{[3,n]}]$
2. Insert a 0-bit in the second position of the vectors from the resulting spanning set

Note that $a^{[2]} = b^{[2]} = 0$, so the n-bit vectors produced this way exactly span the union of the intervals $[00\hat{a}, 001^{\{n-2\}}]$ and $[100^{\{n-2\}}, 10\hat{b}]$.

2.3.3 $a^{[1,2]} = 01$ and $b^{[1,2]} = 11$

This case is complementary to Case 2.3.2. We reduce the instance $[01\hat{a}, 11\hat{b}]$ to the instance $[00\overline{\hat{b}}, 10\overline{\hat{a}}] = [\overline{b}, \overline{a}]$. By Lemma 2.2.3, complementing all the ternary vectors in a minimum spanning set of $[\overline{b}, \overline{a}]$ yields a minimum spanning set of $[a, b]$.

2.3.4 $a^{[1,2]} = 00$ and $b^{[1,2]} = 11$

Let j be maximal such that $a^{[1,j]} = 0^{\{j\}}$ and $b^{[1,j]} = 1^{\{j\}}$. Since $a > 0^{\{n\}}$ (or $b < 1^{\{n\}}$), necessarily $j < n$.

The number j gives us three sub-intervals to span:

- $[a, 0^{\{j\}}1^{\{n-j\}}]$
- $[0^{\{j-1\}}10^{\{n-j\}}, 1^{\{j-1\}}01^{\{n-j\}}]$
- $[1^{\{j\}}0^{\{n-j\}}, b]$

Note that the second sub-interval contains exactly the numbers that don't start with either j zeros or j ones, and as such can be spanned by j ternary vectors. We construct $T_1, \ldots, T_j$ that span the second sub-interval by appending $\phi^{\{n-j\}}$ to each of the j cyclic shifts of $01\phi^{\{j-2\}}$. Thus, $T_1 = 01\phi^{\{j-2\}}\phi^{\{n-j\}}$, ..., $T_i = \phi^{\{i-1\}}01\phi^{\{j-1-i\}}\phi^{\{n-j\}}$ (for $i \in \{1, \ldots, j-1\}$), ..., $T_{j-1} = \phi^{\{j-2\}}01\phi^{\{n-j\}}$, $T_j = 1\phi^{\{j-2\}}0\phi^{\{n-j\}}$.

The other two sub-intervals are spanned recursively. Let $a'' = a^{[j,n]}$ and $b'' = b^{[j,n]}$. Note that $a''^{[1]} = 0$ and $b''^{[1]} = 1$. Note that $|a''| = |b''| = n - j + 1$. Since $j \geq 2$, $n - j + 1 < n$. Let $\mathcal{T}''$ be the spanning set of $[a'', b'']$ computed recursively.

The spanning set of $[a, b]$ will be computed from the vectors $T_1, \ldots, T_j$ and $\mathcal{T}''$ based on the relation between $(n-j)$-bit suffixes of a and b. Let $a' = a^{[j+1,n]}$ and $b' = b^{[j+1,n]}$.

$b' < a' - 1$

In this case, the spanning set of $[a, b]$ consists of the vectors $T_1, \ldots, T_j$ and vectors obtained by prepending each vector from $\mathcal{T}''$ with $\phi^{\{j-1\}}$.

$b' \geq a' - 1$

In this case, the spanning set of $[a, b]$ consists of the vectors $T_1, \ldots, T_{j-1}$ (omitting T_j) and a set $\mathcal{T}'$ which is derived from $\mathcal{T}''$ in the following way:

$$\begin{aligned}\mathcal{T}' = &\{\phi^{\{j-1\}}T | T \in \mathcal{T}'' \text{ and } T^{[1]} \in \{0, \phi\}\} \\ &\cup \{1\phi^{\{j-1\}}T^{[2,n-j+1]} | T \in \mathcal{T}'' \text{ and } T^{[1]} = 1\}\end{aligned}$$

2.4 Discussion

Schieber et al. implicitly use orthogonal sets to prove the optimality of the spanning sets constructed using the algorithm shown in this chapter [8, Theorem 3]. They show a set with the properties of an orthogonal set for each of the cases. This implies the following observation:

Observation 2.4.1. *Every* 1*-interval function is coverable.*

We will not give a detailed proof of Observation 2.4.1 in the general case (Section 2.3). We have proved it in the prefix and suffix case (Corollary 2.2.1).

For the sake of completeness, we note that Schieber et al. also showed a polynomial algorithm for the problem of finding a minimum disjoint spanning set of a 1-interval function [8, Section 4] (see Problem 1.2). Since none of our results uses this algorithm, we shall omit its description.

3. 2-interval functions

In this chapter we will revise the existing results specific to 2-interval functions (Dubovský [4] provides a more detailed overview). We will also introduce a simplified minimization algorithm for 2-switch 2-interval functions and generalize the non-coverability property to all classes of l-switch functions for $l \geq 3$.

Dubovský identified several classes of 2-interval functions [4, p. 5] based on the number of switches, although he did not use the term "switch". Recall that an l-switch function has exactly l input vectors x such that $f(x) \neq f(x+1)$. It is easy to see that a 2-interval function can only have 2, 3 or 4 switches. For 2-switch functions Dubovský proposed a polynomial minimization algorithm while for 3- and 4-switch functions he proposed a polynomial 2-approximation algorithm.

In Section 3.1 we will present a simplified minimization algorithm for 2-switch 2-interval functions. In Section 3.2 we will show that functions with 3 and more switches are not coverable in general.

3.1 2-switch 2-interval functions

In this section we will introduce a polynomial algorithm that solves the following problem:

Problem 3.1 (Minimization of 2-switch 2-interval functions)**.** Given a 2-switch 2-interval function f, find a minimum spanning set of f.

To relate this problem to Problem 1.1, note that since every 2-switch 2-interval function is proper 2-interval, any algorithm that solves the problem *MIN-INT*(2) also solves Problem 3.1.

Let f be a 2-switch 2-interval Boolean function. Dubovský described a polynomial minimization algorithm for this class of functions.[1] The algorithm in Dubovský [4, p. 17] follows the approach from Schieber et al. [8] and relies on a quite complicated and technically involved case analysis. Here we will show that we can find a minimum spanning set of a given 2-switch 2-interval function f by reducing f to a 1-interval function f'. The 1-interval function f' is then spanned optimally using the polynomial algorithm shown in Chapter 2. This approach is much simpler than the case analysis in Dubovský [4].

A 2-switch 2-interval function f necessarily has $f(0^{\{n\}}) = f(1^{\{n\}}) = 1$ and thus there are two numbers b_1 and a_2 such that $f = f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$. Since f is 2-switch, there must be at least one false point between the true points b_1 and a_2 and thus $b_1 < a_2 - 1$ (f is *proper* 2-interval).

Note that f is a negation of the 2-switch 1-interval function $f^n_{[b_1+1,a_2-1]}$. We can also say that f is a *false point* 1*-interval function* as the false points of f form a single interval.

3.1.1 Description of the algorithm

Input A pair of n-bit numbers b_1, a_2 that satisfy the inequality $b_1 < a_2 - 1$.

[1] This definition corresponds to class A_0 in Dubovský's classification [4, p. 5].

The numbers b_1 and a_2 are two of the four endpoints of the 2-switch proper 2-interval function $f^n_{[0\{n\},b_1],[a_2,1\{n\}]} = f$.

Output A spanning set of f. We will denote the output set with $\mathcal{T}$.

Procedure The algoritm constructs $\mathcal{T}$ from two partial spanning sets, $\mathcal{T}_{out}$ and $\mathcal{T}_{in}$.

Let c be the longest common prefix of b_1 and a_2 and let j be its length (thus $b_1^{[1,j]} = a_2^{[1,j]} = c$). Since $b_1 < a_2$, necessarily $j < n$.

We will split the true points of f into two sets S_{out} and S_{in} based on the j-bit prefix. We will span the sets S_{out} and S_{in} separately.

Let S_{out} be the set of all true points of f that do not start with the prefix c. Note that all vectors that do not start with the prefix c are true points of f, since each of them is either smaller than b_1 or larger than a_2. The set S_{out} then contains precisely all the n-bit binary vectors that are strictly smaller than $c0^{\{n-j\}}$ or strictly larger than $c1^{\{n-j\}}$:

$$\begin{aligned} S_{out} &= \{x \in \{0,1\}^n | f(x) = 1 \text{ and } x^{[1,j]} \neq c\} \\ &= \{x \in \{0,1\}^n | x^{[1,j]} \neq c\} \\ &= \{x \in \{0,1\}^n | x < c0^{\{n-j\}} \text{ or } x > c1^{\{n-j\}}\} \end{aligned}$$

We get the set $\mathcal{T}_{out}$ that spans exactly S_{out} in the following way:

$$\mathcal{T}_{out} = \{c^{[1,o-1]}\overline{c^{[o]}}\phi^{\{n-o\}} | o \in \{1, \ldots, j\}\}$$

We still need to span all the true points of f that start with the prefix c. Let S_{in} denote the set of these points:

$$\begin{aligned} S_{in} &= \{x \in \{0,1\}^n | f(x) = 1 \text{ and } x^{[1,j]} = c\} \\ &= \{x \in \{0,1\}^n | (x \leq b_1 \text{ or } x \geq a_2) \text{ and } x^{[1,j]} = c\} \end{aligned}$$

Since $b_1 < a_2$, necessarily $b_1^{[j+1]} = 0$ and $a_2^{[j+1]} = 1$. Let $b'_1 = b_1^{[j+2,n]}$ and $a'_2 = a_2^{[j+2,n]}$. Note that $b_1 = c0b'_1$ and $a_2 = c1a'_2$.

Let us span the 1-interval function $f' = f^{n-j}_{[0a'_2,1b'_1]}$ optimally using the procedure shown in Chapter 2. Let us denote the resulting minimum spanning set of f' with $\mathcal{T}_{[0a'_2,1b'_1]}$. We can use this set to span S_{in}:

$$\mathcal{T}_{in} = \{c\overline{T^{[1]}}T^{[2,n-j]} | T \in \mathcal{T}_{[0a'_2,1b'_1]}\}$$

Having constructed both of the sets $\mathcal{T}_{out}$ and $\mathcal{T}_{in}$, the algorithm returns their union $\mathcal{T} = \mathcal{T}_{out} \cup \mathcal{T}_{in}$.

3.1.2 Feasibility

Theorem 3.1.1. *$\mathcal{T}$ spans exactly $f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$.*

We will separately prove that $\mathcal{T}_{out}$ spans exactly S_{out} and that $\mathcal{T}_{in}$ spans exactly S_{in}. Since $\mathcal{T} = \mathcal{T}_{out} \cup \mathcal{T}_{in}$ and $TP(f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}) = S_{out} \cup S_{in}$, the correctness of Theorem 3.1.1 will follow.

Lemma 3.1.1. *$\mathcal{T}_{out}$ spans exactly S_{out}.*

Proof. In order to show that $span(\mathcal{T}_{out}) = S_{out}$, we will prove the two inclusions separately.

$span(\mathcal{T}_{out}) \subseteq S_{out}$ Let $c^{[1,o-1]}\overline{c^{[o]}}\phi^{\{n-o\}} \in \mathcal{T}_{out}$ span x. Since the j-bit prefix of x differs from c, we conclude that $x \in S_{out}$.

$span(\mathcal{T}_{out}) \supseteq S_{out}$ Let $x \in S_{out}$ such that $x < c0^{\{n-j\}}$. Recall that by Observation 1.2.1, there must be binary vectors $\hat{c}$, x' and c' such that $x = \hat{c}0x'$ and $c0^{\{n-j\}} = \hat{c}1c'$ (namely $\hat{c}$ is a proper prefix of c). Let $o = |\hat{c}| + 1$. It is easy to see that the vector $\hat{c}0\phi^{\{n-o\}} \in \mathcal{T}_{out}$ spans x. If $x > c1^{\{n-j\}}$, a symmetric argument shows a vector in $\mathcal{T}_{out}$ that spans x. We conclude that $\mathcal{T}_{out}$ spans x.

□

Lemma 3.1.2. *$\mathcal{T}_{in}$ spans exactly S_{in}.*

Proof. Again we will prove the two inclusions separately.

1. $span(\mathcal{T}_{in}) \subseteq S_{in}$:

   Let $c\overline{T^{[1]}}T^{[2,n-j]} \in \mathcal{T}_{in}$ span x. Clearly x starts with the prefix c. Let $x = c\alpha x'$ where $\alpha \in \{0,1\}$. Without loss of generality let $\alpha = 0$ (the complementary case is symmetric).

   Since $c\overline{T^{[1]}}T^{[2,n-j]}$ spans $x = c0x'$, T spans $1x'$. Since $T \in \mathcal{T}_{[0a'_2,1b'_1]}$, we have $1x' \in [0a'_2, 1b'_1]$ and especially $1x' \leq 1b'_1$. It follows that $x' \leq b'_1$ and $x = c0x' \leq c0b'_1 = b_1$.

2. $span(\mathcal{T}_{in}) \supseteq S_{in}$:

   Let $x \in S_{in}$. This means that $x^{[1,j]} = c$, and $x \leq b_1$ or $x \geq a_2$. Without loss of generality let us assume that $x \leq b_1$, in particular $x^{[j+1]} = 0$ (the complementary case is symmetric), and let $x' = x^{[j+2,n]}$ (that is $x = c0x'$).

   It follows that $x' \leq b'_1$ and by extension $1x' \leq 1b'_1$. Trivially $1x' \geq 0a'_2$.

   Since $1x' \in [0a'_2, 1b'_1]$, there must be some $T \in \mathcal{T}_{[0a'_2,1b'_1]}$ that spans $1x'$. The corresponding $c\overline{T^{[1]}}T^{[2,n-j]} \in \mathcal{T}_{in}$ clearly spans $x = c0x'$.

□

We conclude that $\mathcal{T}$ spans exactly $f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$.

3.1.3 Optimality

To show that $\mathcal{T}$ is a minimum spanning set of $f = f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$, we will construct an orthogonal set V of f of size $|\mathcal{T}|$. By Theorem 1.6.1 this proves that the spanning set $\mathcal{T}$ is minimum.

The set V consists of three subsets:

- $V_{in} = \{c\overline{v^{[1]}}v^{[2,n-j]} | v \in V_{[0a'_2,1b'_1]}\}$, where $V_{[0a'_2,1b'_1]}$ is an orthogonal set of $f_{[0a'_2,1b'_1]}$ of size $|\mathcal{T}_{[0a'_2,1b'_1]}| = |\mathcal{T}_{in}|$. Such set exists because all 1-interval functions are coverable (see Observation 2.4.1).
- $V_0 = \{c^{[1,o-1]}0c^{[o+1,j]}(b_1+1)^{[j+1,n]} | c^{[o]} = 1\}$
- $V_1 = \{c^{[1,o-1]}1c^{[o+1,j]}(a_2-1)^{[j+1,n]} | c^{[o]} = 0\}$

Note that the vector $c^{[1,o-1]}0c^{[o+1,j]}(b_1+1)^{[j+1,n]} \in V_0$ (for any o such that $c^{[o]} = 1$) is equal to $(b_1+1)^{[1,o-1]}0(b_1+1)^{[o+1,n]}$ because $b_1^{[j+1]} = 0$, and by extension $(b_1+1)^{[1,j]} = b_1^{[1,j]} = c$. A symmetric observation can be made for any vector $v_1 \in V_1$ where we get that $v_1 = (a_2-1)^{[1,o-1]}1(a_2-1)^{[o+1,n]}$ for some o such that $c^{[o]} = 0$.

We know that $|V_{in}| = |\mathcal{T}_{in}|$ by definition of V_{in}. Since there is one vector in V_0 for each 1 bit of c and one vector in V_1 for each 0 bit of c, $|V_0| + |V_1| = |c| = j = |\mathcal{T}_{out}|$.

To see that the sets V_{in}, V_0 and V_1 are pairwise disjoint, consider any $v_{in} \in V_{in}$, $v_0 \in V_0$ and $v_1 \in V_1$. The j-prefix of both v_0 and v_1 differs from c, so $v_0 \neq v_{in}$ and $v_1 \neq v_{in}$. If $v_0^{[1,j]} = c^{[1,o-1]}0c^{[o+1,j]}$, then v_0 differs from v_1 in the o-th position. We conclude that $|V| = |V_{in}| + |V_0| + |V_1|$. Since $|V_{in}| = |\mathcal{T}_{in}|$ and $|V_0| + |V_1| = |\mathcal{T}_{out}|$, $|V| = |\mathcal{T}|$.

We still need to prove that V is orthogonal with respect to f. Recall that by Definition 1.6.1, the set V is orthogonal if and only if it only contains true points and every pair of vectors in V is orthogonal.

Lemma 3.1.3. $V \subseteq TP(f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]})$

Proof. It is easy to see that $v_0 \leq b_1$ and $v_1 \geq a_2$ for any $v_0 \in V_0$ and $v_1 \in V_1$.

Let $v_{in} \in V_{in}$. Let v be the corresponding vector in $V_{[0a'_2,1b'_1]}$ ($v_{in} = c\overline{v^{[1]}}v^{[2,n-j]}$). Since $V_{[0a'_2,1b'_1]}$ is orthogonal with respect to $f_{[0a'_2,1b'_1]}$, necessarily $v \geq 0a'_2$ and $v \leq 1b'_1$. Without loss of generality let $v^{[1]} = 0$ (the complementary case is symmetric). Since $v = 0v^{[2,n-j]} \geq 0a'_2$, $v_{in} = c1v^{[2,n-j]} \geq c1a'_2 = a_2$.

We conclude that all the vectors in V are true points of $f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$. □

Lemma 3.1.4. *Let $x, y \in V$, $x \neq y$. Then x and y are orthogonal with respect to $f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$.*

Proof. We need to show that x and y cannot be spanned by a single ternary vector. Let T be any ternary vector that spans both x and y. We shall consider all six possible combinations one by one:

- $x, y \in V_{in}$: Let $x = c\overline{v_x^{[1]}}v_x^{[2,n-j]}$ and $y = c\overline{v_y^{[1]}}v_y^{[2,n-j]}$. The vectors v_x and v_y are orthogonal with respect to $f_{[0a'_2,1b'_1]}$ by definition of $V_{[0a'_2,1b'_1]}$.

Let $\hat{T} = \overline{T^{[j+1]}}T^{[j+2,n]}$. Since v_x and v_y are orthogonal with respect to $f_{[0a'_2,1b'_1]}$ and since $\hat{T}$ spans both v_x and v_y, let $\hat{z}$ be a false point of $f_{[0a'_2,1b'_1]}$ that is spanned by $\hat{T}$. Without loss of generality let $\hat{z}^{[1]} = 0$ (the complementary case is symmetric). Since $\hat{z}$ is a false point of $f_{[0a'_2,1b'_1]}$ and $\hat{z}^{[1]} = 0$, necessarily $\hat{z} = 0\hat{z}^{[2,n-j]} < 0a'_2$ and by extension $\hat{z}^{[2,n-j]} < a'_2$.

Let $z = c\overline{\hat{z}^{[1]}}\hat{z}^{[2,n-j]} = c1\hat{z}^{[2,n-j]}$. To see that z is a false point of the 2-interval function $f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$, note that $z > b_1$ since $z^{[j+1]} = 1$ and $z < a_2$ since $z^{[j+2,n]} < a'_2$. To see that T spans z, note that $T^{[1,j]}$ spans $x^{[1,j]} = c = z^{[1,j]}$ and $T^{[j+1,n]} = \overline{\hat{T}^{[1]}}\hat{T}^{[2,n-j]}$ spans $\overline{\hat{z}^{[1]}}\hat{z}^{[2,n-j]} = z^{[j+1,n]}$.

- $x, y \in V_0$: T spans the false point $b_1 + 1$.
- $x, y \in V_1$: T spans the false point $a_2 - 1$.
- $x \in V_{in}, y \in V_0$: T spans the false point $b_1 + 1$.
- $x \in V_{in}, y \in V_1$: T spans the false point $a_2 - 1$.
- $x \in V_0, y \in V_1$: T spans the false point $b_1 + 1$.

We have shown that no matter which combination of different vectors x, y from V and a ternary vector that spans both x and y we choose, the ternary vector necessarily spans some false point of $f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$. □

We conclude that V is orthogonal with respect to $f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$. Since $|V| = |\mathcal{T}|$, by Theorem 1.6.1 $\mathcal{T}$ is a minimum spanning set of $f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$.

Corollary 3.1.1. *Every 2-switch 2-interval function is coverable.*

Since every 2-switch function is either 1- or 2-interval (see Example 1.1) and every 1-switch function is 1-interval, we collect the results from Observation 2.4.1 and Corollary 3.1.1 in the following statement:

Corollary 3.1.2. *Every function with at most 2 switches is coverable.*

3.2 Functions with 3 and more switches

The situation changes in 3-switch functions. In 1-switch (prefix and suffix) and 2-switch (1-interval and 2-interval with extreme outer endpoints) functions, we managed to efficiently find minimum spanning sets. The proofs of optimality of the spanning sets depend on construction of sufficiently large orthogonal sets.[2] The fact that this aproach succeeds depends on the fact that all the functions we have considered so far (1- and 2-switch) are *coverable* (according to Definition 1.6.2).

Dubovský has shown that the 3-switch function $f = f^4_{[0,4],[11,14]}$ is not coverable. The size of a maximum orthogonal set of f is 4 while the size of a minimum

[2]While we have shown an orthogonal set explicitly in the cases of prefix functions (Section 2.2.2) and 2-switch 2-interval functions (Section 3.1), the correspondence between suffix and prefix intervals (Section 2.2.1) clearly preserves orthogonality of sets and Schieber et al. [8] use orthogonal sets implicitly for general 1-interval functions.

spanning set of f is 5 [4, p. 32]. This shows that 3-switch functions are not coverable in general. Dubovský proved these bounds on sizes of orthogonal sets and spanning sets of f by exhaustion using software tools. We will generalize the result for functions with larger number of switches.

Lemma 3.2.1. *If there is an l-switch function that is not coverable, then there is also an $(l+1)$-switch function that is not coverable.*

Proof. Let f be an n-ary l-switch function that is not coverable. We will construct an $(n+1)$-ary $(l+1)$-switch function f' that is not coverable. We shall distinguish two cases depending on the value of f on $1^{\{n\}}$.

1. $f(1^{\{n\}}) = 1$:

   Let f' be an $(n+1)$-ary function defined in the following way:

   $$f'(x') = \begin{cases} f(x'^{[2,n+1]}) & \text{if } x'^{[1]} = 0 \\ 0 & \text{if } x'^{[1]} = 1 \end{cases}$$

   Since f is l-switch, there are exactly l vectors x of length n ($x < 1^{\{n\}}$) such that $f(x) \neq f(x+1)$. Prepending a 0 to each of these vectors, we get l vectors x' of length $n+1$ such that $f'(x') \neq f'(x'+1)$. The $(l+1)$-st switch vector of f' is $01^{\{n\}}$, since $f'(01^{\{n\}}) = 1$ and $f'(10^{\{n\}}) = 0$. On the other hand, observe that all switch vectors of f' correspond to switch vectors of f with the single exception of $01^{\{n\}}$. Thus we have shown that f' is $(l+1)$-switch.

   There is a size preserving 1-1 correspondence between the spanning sets of f and f'. A spanning set $\mathcal{T}$ of f corresponds to the spanning set $0\mathcal{T} = \{0T | T \in \mathcal{T}\}$ of f'. The same holds for orthogonal sets. This proves that if f is not coverable, neither is f'.

2. $f(1^{\{n\}}) = 0$:

   Let f' be an $(n+1)$-ary function defined in the following way:

   $$f'(x') = \begin{cases} f(x'^{[2,n+1]}) & \text{if } x'^{[1]} = 0 \\ 0 & \text{if } 10^{\{n\}} \leq x' \leq 1^{\{n\}}0 \\ 1 & \text{if } x' = 1^{\{n+1\}} \end{cases}$$

   Again, the l switches of f translate to l switches of f' by prepending a 0. The $(l+1)$-st switch is $1^{\{n\}}0$ in this case.

   To show that f' is not coverable, we will prove bounds on the size of its minimum spanning set and maximum orthogonal set.

   $dnf(f') \geq dnf(f) + 1$ We need a dedicated ternary vector to span the true point $1^{\{n+1\}}$ in f'. If a ternary vector spanned both $1^{\{n+1\}}$ and $0x$ for any n-bit x, it would necessarily also span the false point $01^{\{n\}}$.

   If we could span f' with less than $dnf(f) + 1$ vectors, we could span f with less than $dnf(f)$ vectors, since given a spanning set of f',

leaving out the vector that spans the true point $1^{\{n+1\}}$ and removing the leading symbol of the rest (necessarily a 0) gives a spanning set of f.

$ortho(f') \leq ortho(f) + 1$ Let us proceed by contradiction and let V' be an orthogonal set of f' of size $ortho(f) + 2$. Let $V = \{v'^{[2,n+1]} | v' \in V' \text{ and } v'^{[1]} = 0\}$. Since V' only consists of true points of f', there can be at most one vector in V' that starts with a 1. It follows that $|V| \geq |V'| - 1 = ortho(f) + 1$. V is an orthogonal set of f – if $0u$ and $0v$ are orthogonal in f', every ternary vector that spans them must span a false point $0x$. Leaving out the leading symbol preserves the relation, and thus orthogonality as well. We have shown an orthogonal set of f of size at least $ortho(f) + 1$, which contradicts the premise that $ortho(f)$ is the size of a maximum orthogonal set of f.

Since f is not coverable, $ortho(f) < dnf(f)$. Putting the inequalities together:

$$ortho(f') \leq ortho(f) + 1 < dnf(f) + 1 \leq dnf(f')$$

We conclude that f' is an $(l+1)$-switch non-coverable function.

□

We conclude with the following theorem:

Theorem 3.2.1. *For any $l \geq 3$, there is an l-switch non-coverable function. Moreover, the arity of such function can be as small as $l + 1$.*

Proof. Starting with the 4-ary 3-switch function $f^4_{[0,4],[11,14]}$ from Dubovský [4], we can get a non-coverable l-switch function for any $l \geq 3$ by using Lemma 3.2.1 inductively. The arity of $f^4_{[0,4],[11,14]}$ is 4 and every time we increment l in Lemma 3.2.1, we increase the arity by 1. □

This result indicates that if we consider an algorithm that minimizes the representations of all l-switch functions for any $l \geq 3$, then we have to use a different tool than orthogonal sets to show that the algorithm really finds an optimal spanning set.

4. Suffix-prefix decomposition algorithm for k-interval functions

In this chapter we will show a simple polynomial $2k$-approximation algorithm for Problem 1.1, that is *MIN-INT*(k), for every $k \geq 0$. We shall assume that the function f is represented by the endpoints of its intervals. The algorithm we will describe is a straightforward extension of Schieber et al.'s suffix-prefix approximation algorithm for disjoint spanning sets of 1-interval functions [8, Section 6].

4.1 Description of the algorithm

Algorithm 4.1 (Suffix-prefix decomposition)**.**

Input n-bit binary vectors $a_1, b_1, \ldots, a_k, b_k$ for $n \geq 0$ and $k \geq 0$ that satisfy the inequalities $0^{\{n\}} \leq a_1$, $a_1 \leq b_1$, $b_1 < a_2 - 1$, $\ldots$, $a_i \leq b_i$, $b_i < a_{i+1} - 1$ (for $i \in \{1, \ldots, k-1\}$), $\ldots$, $b_{k-1} < a_k - 1$, $a_k \leq b_k$, $b_k \leq 1^{\{n\}}$.

Such numbers are the endpoints of the intervals of the proper k-interval Boolean function $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$.

Output A spanning set of $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$.

Procedure The algorithm spans each of the intervals $[a_i, b_i]$ separately. The set that spans $[a_i, b_i]$ will be denoted $\mathcal{T}_i$, and the output set will be denoted $\mathcal{T} = \bigcup_{i=1}^{k} \mathcal{T}_i$.

Let us consider the interval $[a_i, b_i]$. Let c be the longest common prefix of a_i and b_i and let j be its length.

If $j = n$, that is $c = a_i = b_i$, $\mathcal{T}_i$ consists of the single vector $c = a_i = b_i$.

We are left with the situation $j < n$.

Note that $a_i^{[j+1]} = 0$ and $b_i^{[j+1]} = 1$, since $a_i \leq b_i$ and $a_i \neq b_i$. Now let $a' = a_i^{[j+2,n]}$ and $b' = b_i^{[j+2,n]}$ ($a_i = c0a'$ and $b_i = c1b'$). Optimally span the suffix interval $[a', 1^{\{n-j-1\}}]$ and the prefix interval $[0^{\{n-j-1\}}, b']$ using the algorithm introduced in Section 2.2 and producing spanning sets $\mathcal{T}'_{i,0}$ and $\mathcal{T}'_{i,1}$ respectively. Now simply prepend $a_i^{[1,j+1]} = c0$ to all vectors in $\mathcal{T}'_{i,0}$ and $b_i^{[1,j+1]} = c1$ to all vectors in $\mathcal{T}'_{i,1}$, producing $\mathcal{T}_{i,0}$ and $\mathcal{T}_{i,1}$ respectively. The spanning set of $[a_i, b_i]$ is then $\mathcal{T}_i = \mathcal{T}_{i,0} \cup \mathcal{T}_{i,1}$.

4.2 Feasibility

Theorem 4.2.1. *The algorithm spans exactly* $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$.

Proof. We need to show that for each i, $\mathcal{T}_i$ spans exactly $[a_i, b_i]$. This is obvious in case $j = n$.

In the remaining case $j < n$, observe that $0\mathcal{T}'_{i,0} = \{0T | T \in \mathcal{T}'_{i,0}\}$ spans exactly the interval $[0a', 01^{\{n-j-1\}}]$ and $1\mathcal{T}'_{i,1}$ spans exactly the interval $[10^{\{n-j-1\}}, 1b']$. Since $10^{\{n-j-1\}} = 01^{\{n-j-1\}} + 1$, the union of these sets spans exactly the interval $[0a', 1b'] = [a_i^{[j+1,n]}, b_i^{[j+1,n]}]$. Prepending the common prefix c to $0\mathcal{T}'_{i,0} \cup 1\mathcal{T}'_{i,1}$ preserves the spanning to $[a_i, b_i]$ and matches the operation performed in the algorithm to produce $\mathcal{T}_i$.

The union of such $\mathcal{T}_i$s clearly spans exactly the union of the intervals. □

4.3 Approximation ratio

Let F be a class of Boolean functions. We shall denote the worst case approximation ratio of Algorithm 4.1 on functions from class F with $approx_{SPD}(F)$:

$$approx_{SPD}(F) = \sup_{f \in F} \frac{spd(f)}{dnf(f)}$$

where $spd(f)$ is the size of the spanning set of f returned by the suffix-prefix decomposition algorithm.

We will show for every $k \geq 1$ that $approx_{SPD}(INT(k)) = 2k$ where $INT(k)$ is the class of all k-interval Boolean functions (see Definition 1.7.3).

Theorem 4.3.1. *Let $f = f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$ be a k-interval function for $k \geq 1$. Then $spd(f) \leq 2k \cdot dnf(f)$.*

Proof. To prove the statement, we will show an orthogonal set V of the function $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$ of size $|V| \geq \frac{spd(f)}{2k}$. By Theorem 1.6.1, the existence of such orthogonal set gives a lower bound $|V|$ on $dnf(f)$.

Let $\mathcal{T} = \bigcup_{i=1}^{k} \mathcal{T}_i$ be the spanning set of f returned by the suffix-prefix decomposition algorithm ($|\mathcal{T}| = spd(f)$).

Let us first consider the case that $|\mathcal{T}_i| \leq 2$ for every i. Since $|\mathcal{T}_i| \leq 2$ for every $i \in \{1, \ldots, k\}$, clearly $|\mathcal{T}| \leq 2k$. Since $k \geq 1$, necessarily $dnf(f) \geq 1$. It is then easy to see that $|\mathcal{T}| \leq 2k \leq 2k \cdot dnf(f)$.

We are now left with the case that $|\mathcal{T}_i| > 2$ for some i. Necessarily $\mathcal{T}_i = \mathcal{T}_{i,0} \cup \mathcal{T}_{i,1}$ for such i. One of the sets $\mathcal{T}_{i,0}$ and $\mathcal{T}_{i,1}$ must contain at least 2 vectors because otherwise $|\mathcal{T}_i| = 2$.

Let $\mathcal{T}_{i,\alpha}$ ($i \in \{1, \ldots, k\}, \alpha \in \{0, 1\}$) be the largest of the partial spanning sets of this type (that is suffix or prefix) produced during the course of the algorithm. Note that $|\mathcal{T}_{i,\alpha}| \geq 2$.

Let c be the longest common prefix of a_i and b_i and let j be its length. Without loss of generality, let $\alpha = 1$ (the complementary case $\alpha = 0$ is symmetric). The set $\mathcal{T}'_{i,1}$ that spans exactly the interval $[0^{\{n-j-1\}}, b' = b_i^{[j+2,n]}]$ was in this case obtained using the prefix algorithm introduced in Section 2.2.2. Note that $b' < 1^{\{n-j-1\}}$ because otherwise $|\mathcal{T}_{i,1}| = 1$. In the proof of optimality of the prefix algorithm (see Theorem 2.2.2), we showed an orthogonal set that matches the size of the spanning set produced by the algorithm. Let us denote this orthogonal set of $f^{n-j-1}_{[0^{\{n-j-1\}},b']}$ with $V'_{i,1}$.

Let $V = \{c1v | v \in V'_{i,1}\}$. We claim that V is an orthogonal set of $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$.

To see that every vector $c1v \in V$ is a true point, note that trivially $c1v \geq c0a' = a_i$ and since $V'_{i,1}$ only contains true points of $f^{n-j-1}_{[0^{\{n-j-1\}},b']}$, necessarily $v \leq b'$ and by extension $c1v \leq c1b' = b_i$.

Let $c1x, c1y \in V$ and $x \neq y$. Recall that by Corollary 2.2.2 every pair of distinct vectors in $V'_{i,1}$ is separated by the false point $b'+1$ of $f^{n-j-1}_{[0^{\{n-j-1\}},b']}$ (see Definition 1.6.3). Since the vector $b'+1$ separates x and y, the vector $c1(b'+1) = b_i + 1$ separates $c1x$ and $c1y$. To see that $b_i + 1$ is a false point of $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$, note that if $i < k$, then $b_i < a_{i+1} - 1$ by the requirements on input data (we only consider *proper* k-interval functions).

It follows that V is orthogonal with respect to $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$. From Theorem 1.6.1 we get $dnf(f) \geq |V|$.

Since $\mathcal{T}_{i,\alpha}$ is the largest of the at most $2k$ spanning sub-sets of this type and the spanning sets of the other types are of size 1, necessarily $spd(f) = |\mathcal{T}| \leq 2k|\mathcal{T}_{i,\alpha}| = 2k|V| \leq 2k \cdot dnf(f)$. □

Note that the spanning set returned by the algorithm is disjoint. This makes the algorithm $2k$-approximation for the disjoint case as well:

Theorem 4.3.2. *Let $\mathcal{T}_{disjoint}$ be an optimal* disjoint *spanning set of the function $f = f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$ and let $\mathcal{T}$ be the (disjoint) spanning set returned by our algorithm. We claim that:*

$$|\mathcal{T}| \leq 2k|\mathcal{T}_{disjoint}|$$

Proof. Since every disjoint spanning set is a spanning set in general, necessarily $dnf(f) \leq |\mathcal{T}_{disjoint}|$.

Using Theorem 4.3.1 we get:

$$|\mathcal{T}| \leq 2k \cdot dnf(f) \leq 2k|\mathcal{T}_{disjoint}|$$

□

Theorem 4.3.3. *The approximation ratio of $2k$ is tight.*

Proof. For every k that is a power of 2 we will show a k-interval function f such that $|\mathcal{T}| = 2k \cdot dnf(f)$, where $\mathcal{T}$ is the spanning set of f returned by the suffix-prefix decomposition algorithm.

Let $k = 2^{n_k}$. Let P be all the n_k-bit numbers, that is $P = \{0,1\}^{n_k}$. Note that $|P| = 2^{n_k} = k$. Let $n = n_k + 3$.

For each $p \in P$, we define the interval $[a_p, b_p]$ by appending certain 3-bit suffixes to p:

- $a_p = p000$
- $b_p = p011$

The first appended bit (0 for a_p and 0 for b_p) ensures that there is at least one false point between any pair of intervals defined this way (all the points that have a 1 in their (n_k+1)-st position are false points).

The remaining pair of appended bits (00 for a_p and 11 for b_p) ensures that $a_p^{[1,n-1]} \neq b_p^{[1,n-1]}$, so the algorithm will span the sub-intervals $[a_p = p000, p001]$ and $[p010, b_p = p011]$ separately, producing at least 2 vectors to span $[a_p, b_p]$. The

configuration also enables the interval $[a_p, b_p]$ to be spanned by the single ternary vector $p0\phi^{\{2\}}$.

Thus we have $k = 2^{n_k}$ intervals none of which intersect or are adjoint. Let f be the proper k-interval function defined by the intervals $[a_p, b_p]$ for $p \in P$. Since $P = \{0,1\}^{n_k}$, we can span all the intervals by the single ternary vector $\phi^{\{n_k\}}0\phi^{\{2\}}$. Clearly $dnf(f) = 1$.

However, the approximation algorithm uses $2k$ vectors to span the intervals, since it spans each of the intervals separately and uses the two vectors $p00\phi, p01\phi$ for each interval.

We get $|\mathcal{T}| = 2k = 2k \cdot dnf(f)$.

Note that the optimal spanning set of f is trivially disjoint, so the ratio is tight in disjoint case as well. □

5. Interval decomposition algorithm for k-interval functions

The $2k$-approximation suffix-prefix decomposition algorithm introduced in Chapter 4 can be naturally improved by spanning each of the k intervals of the given k-interval function optimally using the minimization algorithm introduced in Chapter 2 (originally in Schieber et al. [8]). We will call this improved algorithm "interval decomposition algorithm".

Since the interval decomposition algorithm clearly performs at least as well as the suffix-prefix decomposition algorithm, it is $2k$-approximation as well (see Theorem 4.3.1). Dubovský has managed to prove that on 2-interval functions, the interval decomposition algorithm has an approximation ratio 2 [4, p. 39], while the suffix-prefix decomposition algorithm has a tight approximation ratio 4 on 2-interval functions (see Theorem 4.3.3). This might lead us to expect that the interval decomposition algorithm performs significantly better in general. However, we will show in this chapter that its approximation ratio is close to $2k$ for large values of k.

5.1 Description of the algorithm

Algorithm 5.1 (Interval decomposition)**.**

Input n-bit binary vectors $a_1, b_1, \ldots, a_k, b_k$ for $n \geq 0$ and $k \geq 0$ that satisfy the inequalities $0^{\{n\}} \leq a_1$, $a_1 \leq b_1$, $b_1 < a_2 - 1$, $\ldots$, $a_i \leq b_i$, $b_i < a_{i+1} - 1$ (for $i \in \{1, \ldots, k-1\}$), $\ldots$, $b_{k-1} < a_k - 1$, $a_k \leq b_k$, $b_k \leq 1^{\{n\}}$.

Such numbers are the endpoints of the intervals of the proper k-interval Boolean function $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$.

Output A spanning set of $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$.

Procedure The algorithm spans each of the k intervals $[a_i, b_i]$ separately. For every i, let $\mathcal{T}_i$ be the spanning set of the interval $[a_i, b_i]$ returned by the 1-interval optimization algorithm introduced in Chapter 2. The algorithm returns $\mathcal{T} = \bigcup_{i=1}^{k} \mathcal{T}_i$.

Note that the spanning set constructed by the interval decomposition algorithm need not be disjoint (unlike the spanning set constructed by the suffix-prefix decomposition algorithm). If we used Schieber et al.'s 1-interval disjoint spanning set minimization algorithm [8], we would obtain a disjoint spanning set. We will not analyze the approximation ratio in this case.

5.2 Approximation ratio

Similarly to Section 4.3, let F be a class of Boolean functions. We shall denote the worst case approximation ratio of the interval decomposition algorithm on

functions from class F with $approx_{ID}(F)$:

$$approx_{ID}(F) = \sup_{f \in F} \frac{id(f)}{dnf(f)}$$

where $id(f)$ is the size of the spanning set of f returned by the interval decomposition algorithm.

We will analyze the approximation ratio of Algorithm 5.1 on the class $INT(k)$ of all k-interval Boolean functions (see Definition 1.7.3) for every $k \geq 1$. We will not derive an exact approximation ratio of Algorithm 5.1 on $INT(k)$, instead we will show that $approx_{ID}(INT(k))$ is upper bounded by $2k$ for $k \geq 1$ and we will provide a lower bound $2k-2$ for all values of k that are equal to $2^{l-1}+1$ for some $l \geq 1$.

5.2.1 Upper bound of $2k$

Since the interval decomposition algorithm spans each of the k intervals separately and optimally, while the suffix-prefix decomposition algorithm spans each of them separately and possibly non-optimally, we get an upper bound $2k$ on $approx_{ID}(INT(k))$ by Theorem 4.3.1:

Observation 5.2.1. *Then* $approx_{ID}(INT(k)) \leq approx_{SPD}(INT(k)) = 2k$ *for every* $k \geq 1$.

However, there are functions on which Algorithm 5.1 performs strictly better than Algorithm 4.1. In fact, $approx_{ID}(INT(k))$ is exactly k for $k \in \{1, 2\}$ as will be discussed in Section 5.2.2. Therefore the approximation ratio of $2k$ is not tight in general. The following sections will deal with showing lower bounds on $approx_{ID}(INT(k))$.

5.2.2 Known lower bounds for $k \in \{1, 2\}$

Dubovský has shown that the approximation ratio is strictly smaller than $2k$ for $k = 2$, as the algorithm is 2-approximation in this case [4, p. 39]. Similarly, it is obvious that the algorithm gives an optimal result in case $k = 1$ (which corresponds to the approximation ratio 1).

$$approx_{ID}(INT(1)) = 1 \tag{5.1}$$
$$approx_{ID}(INT(2)) = 2 \tag{5.2}$$

In the rest of this chapter we shall construct a class of k-interval functions which show that the approximation ratio is strictly larger than k for k as small as 3. We will even show that the approximation ratio of Algorithm 5.1 is relatively close to $2k$ for large values of k.

5.2.3 A class of functions achieving the lower bound

In this section we will describe a class of functions on which the interval decomposition algorithm performs particularly badly. For every $l \geq 1$ and $n \geq l+2$, we will construct a proper $(2^{l-1}+1)$-interval n-ary function f_l^n.

Definition 5.2.1 (Difficult function f_l^n)**.** Let $l \geq 1$ denote a parameter that determines the number of intervals of f_l^n. Let $n \geq l+2$ denote the arity of f_l^n (and thus the length of its points, especially the interval endpoints). Let $k = 2^{l-1}+1$ denote the number of intervals of f_l^n.

The endpoints of the k intervals of the function f_l^n shall be denoted as $a_1, b_1, \ldots, a_k, b_k$, that is the i-th interval of f_l^n is $[a_i, b_i]$. All the endpoints $a_1, \ldots, a_k$ have the same suffix $s_a = 0^{\{n-l-1\}}1$ and all the endpoints $b_1, \ldots, b_k$ have the same suffix $s_b = 1^{\{n-l-1\}}0$, both of them having the length $n-l$. Given an index $i \in \{1, \ldots, k\}$ the endpoints a_i and b_i have the l-bit prefix p_{a_i} and p_{b_i} which are defined as follows:

- $p_{a_1} = p_{b_1} = 0^{\{l\}}$
- $p_{a_i} = 2 \cdot i - 3$ for $i \in \{2, \ldots, k\}$
- $p_{b_i} = 2 \cdot i - 2$ for $i \in \{1, \ldots, k-1\}$
- $p_{a_k} = p_{b_k} = 1^{\{l\}}$

To summarize our definition, for a given index $i \in \{1, \ldots, k\}$ we have that $a_i = p_{a_i} s_a$ and $b_i = p_{b_i} s_b$. We will prove that the function $f_l^n = f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$ is a correctly defined proper k-interval function in the observations that follow.

Observation 5.2.2. *$p_{b_i} = p_{a_i} + 1$ for $i \in \{2, \ldots, k-1\}$.*

Proof. The observation is a trivial application of Definition 5.2.1. □

Observation 5.2.3. *$p_{a_i} \leq p_{b_i}$ for $i \in \{1, \ldots, k\}$.*

Proof. The fact that $p_{a_i} \leq p_{b_i}$ for $i \in \{2, \ldots, k-1\}$ follows from Observation 5.2.2. For $i \in \{1, k\}$, we have $p_{a_i} = p_{b_i}$ by Definition 5.2.1. □

Observation 5.2.4. *$p_{a_{i+1}} = p_{b_i} + 1$ for $i \in \{1, \ldots, k-1\}$.*

Proof. We will show this by expanding Definition 5.2.1 separately for various values of i.

- $i \in \{2, \ldots, k-2\}$:
  $p_{a_{i+1}} = 2 \cdot (i+1) - 3 = 2 \cdot i - 1 = (2 \cdot i - 2) + 1 = p_{b_i} + 1$
- $i = 1$:
  $p_{a_2} = 2 \cdot 2 - 3 = 1 = 0 + 1 = p_{b_1} + 1$
- $i = k-1$:

$$\begin{aligned} p_{a_k} &= 1^{\{l\}} = 2^l - 1 = (2 \cdot 2^{l-1} - 2) + 1 \\ &= (2 \cdot ((2^{l-1}+1) - 1) - 2) + 1 = (2 \cdot (k-1) - 2) + 1 = p_{b_{k-1}} + 1 \end{aligned}$$

□

Corollary 5.2.1. *$p_{b_i} < p_{a_{i+1}}$ for $i \in \{1, \ldots, k-1\}$.*

Corollary 5.2.2. *$b_i < a_{i+1}$ for $i \in \{1, \ldots, k-1\}$.*

Observation 5.2.5. *The numbers p_{a_i}, p_{b_i} for $i \in \{1, \ldots, k\}$ are correctly defined l-bit numbers.*

Proof. We need to show that none of the numbers $a_1, b_1, \ldots, a_k, b_k$ is smaller than $0^{\{l\}}$ or larger than $1^{\{l\}}$.

From Observation 5.2.3 and Corollary 5.2.1 it follows that p_{a_1} is the smallest and p_{b_k} is the largest of the numbers $p_{a_1}, p_{b_1}, \ldots, p_{a_k}, p_{b_k}$. From Definition 5.2.1 we see that $p_{a_1} = 0^{\{l\}}$ and $p_{b_k} = 1^{\{l\}}$. □

Observation 5.2.6. *The numbers $a_1, b_1, \ldots, a_k, b_k$ are endpoints of a k-interval function by Definition 1.7.1.*

Proof. Clearly $a_1 \geq 0^{\{n\}}$ and $b_k \leq 1^{\{n\}}$, since each of the numbers consists of an l-bit prefix (see Observation 5.2.5) and an $(n-l)$-bit suffix (see Definition 5.2.1).

The fact that $a_i = p_{a_i} s_a \leq p_{b_i} s_b = b_i$ for $i \in \{1, \ldots, k\}$ follows from Observation 5.2.3 and Definition 5.2.1.

The fact that $b_i < a_{i+1}$ for $i \in \{1, \ldots, k-1\}$ follows from Corollary 5.2.2. □

We conclude that f_l^n is a correctly defined k-interval function. We shall continue to show that f_l^n is actually a *proper* k-interval function by Definition 1.7.2.

Observation 5.2.7. *The vector $a_i - 1$ for $i \in \{1, \ldots, k\}$ is a false point of f_l^n.*

Proof. First note that $a_i - 1 \geq 0$, since $a_i^{[n]} = 1$ by Definition 5.2.1.

Since f_l^n is a correctly defined k-interval function by Observation 5.2.6, the vector $a_i - 1$ can only be a true point if $i \geq 2$ and $a_i - 1 = b_{i-1}$. However, this cannot happen because $(a_i - 1)^{[l+1,n]} = 0^{\{n-l\}} \neq 1^{\{n-l-1\}}0 = b_{i-1}^{[l+1,n]}$ by Definition 5.2.1. □

Corollary 5.2.3. *The function f_l^n is a* proper *k-interval function by Definition 1.7.2.*

We will continue to examine the properties of f_l^n in several observations that will come useful later.

Observation 5.2.8. *p_{b_i} is even for every $i \in \{1, \ldots, k-1\}$.*

Proof. This is easy to see from Definition 5.2.1. □

Observation 5.2.9. *The numbers p_{b_i} for $i \in \{1, \ldots, k-1\}$ are exactly all the l-bit even numbers.*

Proof. There are 2^{l-1} l-bit distinct even numbers and we have defined $k-1 = 2^{l-1}$ l-bit prefixes p_{b_i} for $i \in \{1, \ldots, k-1\}$ each of which is even by Observation 5.2.8. Corollary 5.2.1 and Observation 5.2.3 show that these prefixes are pairwise different. We conclude that the prefixes p_{b_i} for $i \in \{1, \ldots, k-1\}$ exhaust all l-bit even numbers. □

Observation 5.2.10. *Every false point of f_l^n ends with $0^{\{n-l\}}$ or $1^{\{n-l\}}$.*

Proof. Let x be any n-bit binary vector. Let $p_x = x^{[1,l]}$ and $s_x = x^{[l+1,n]}$. We will prove the contrapositive implication, that is if $s_x \notin \{0^{\{n-l\}}, 1^{\{n-l\}}\}$, then x is a true point.

It is easy to see that $s_x \notin \{0^{\{n-l\}}, 1^{\{n-l\}}\}$ if and only if $s_x \geq 0^{\{n-l-1\}}1$ and $s_x \leq 1^{\{n-l-1\}}0$, and that x is a true point if and only if there is some $i \in \{1, \ldots, k\}$ such that $x \in [a_i, b_i]$. We will thus continue to prove the following statement equivalent to Observation 5.2.10:

Claim 5.2.1. *Let $s_x \in [0^{\{n-l-1\}}1, 1^{\{n-l-1\}}0]$. Let $i = \lceil \frac{p_x}{2} \rceil + 1$. We claim that $i \in \{1, \ldots, k\}$ and $x \in [a_i, b_i]$.*

- $i \in \{1, \ldots, k\}$:

  It is easy to see that i defined as $\lceil \frac{p_x}{2} \rceil + 1$ is a whole number greater than or equal to 1. To see that $i \leq k$, recall that $k = 2^{l-1} + 1$ by Definition 5.2.1 and note that $p_x \leq 2^l - 1$ since p_x is an l-bit vector.

$$i = \left\lceil \frac{p_x}{2} \right\rceil + 1 \leq \left\lceil \frac{2^l - 1}{2} \right\rceil + 1 = \left\lceil 2^{l-1} - \frac{1}{2} \right\rceil + 1 = 2^{l-1} + 1 = k$$

- $x \in [a_i, b_i]$:

  We will first show that $p_x \in [p_{a_i}, p_{b_i}]$.

  To see that $p_{a_i} \leq p_x$:

$$\begin{aligned} p_{a_i} \leq p_x &\iff 2 \cdot i - 3 \leq p_x \\ &\iff 2 \cdot \left(\left\lceil \frac{p_x}{2} \right\rceil + 1 \right) - 3 \leq p_x \\ &\iff 2 \cdot \left\lceil \frac{p_x}{2} \right\rceil + 2 - 3 \leq p_x \\ &\iff 2 \cdot \left\lceil \frac{p_x}{2} \right\rceil \leq p_x + 1 \end{aligned} \tag{5.3}$$

  It is easy to see that the inequality (5.3) holds in case p_x is even. In case $p_x = 2r + 1$ for some r:

$$\begin{aligned} 2 \cdot \left\lceil \frac{p_x}{2} \right\rceil = 2 \cdot \left\lceil \frac{2r+1}{2} \right\rceil = 2 \cdot \left\lceil r + \frac{1}{2} \right\rceil = 2 \cdot \left(r + \left\lceil \frac{1}{2} \right\rceil \right) \\ = 2 \cdot (r+1) = 2r + 2 = (2r+1) + 1 = p_x + 1 \end{aligned}$$

  We conclude that $p_{a_i} \leq p_x$.

  To see that $p_{b_i} \geq p_x$:

$$p_{b_i} = 2 \cdot i - 2 = 2 \cdot \left(\left\lceil \frac{p_x}{2} \right\rceil + 1 \right) - 2 = 2 \cdot \left\lceil \frac{p_x}{2} \right\rceil + 2 - 2 = 2 \cdot \left\lceil \frac{p_x}{2} \right\rceil \geq p_x$$

  Since $p_{a_i} \leq p_x$ and $s_a \leq s_x$ (by Definition 5.2.1 $s_a = 0^{\{n-l-1\}}1$, and by the premise of Claim 5.2.1 $s_x \geq 0^{\{n-l-1\}}1$), $a_i = p_{a_i} s_a \leq p_x s_x = x$. A symmetric argument shows that $b_i \geq x$.

We have shown that every point that does not end with $0^{\{n-l\}}$ or $1^{\{n-l\}}$ is a true point of f_l^n. Conversely every false point of f_l^n ends with $0^{\{n-l\}}$ or $1^{\{n-l\}}$. $\square$

Observation 5.2.11. *The true points of f_l^n that end with $0^{\{n-l\}}$ are exactly all the points $p_{b_i}0^{\{n-l\}}$ for $i \in \{2, \ldots, k-1\}$.*

Proof. Let x be any n-bit binary vector. Let $p_x = x^{[1,l]}$ and $s_x = x^{[l+1,n]} = 0^{\{n-l\}}$. We claim that x is a true point of f_l^n if and only if $p_x = p_{b_i}$ for some $i \in \{2, \ldots, k-1\}$.

1. If $f_l^n(x) = 1$, then $p_x = p_{b_i}$ for some $i \in \{2, \ldots, k-1\}$:

   Let x be a true point of f_l^n. Let $[a_i, b_i] = [p_{a_i}s_a, p_{b_i}s_b]$ be the interval of f_l^n that contains x. We claim that $i \in \{2, \ldots, k-1\}$ and $p_x = p_{b_i}$.

   (a) $i \in \{2, \ldots, k-1\}$:
   If $i \in \{1, k\}$, then $p_{a_i} = p_{b_i}$ by Definition 5.2.1. Since $s_a = 0^{\{n-l-1\}}1$ by Definition 5.2.1, the interval $[a_i, b_i]$ does not contain any vector that ends with $0^{\{n-l\}}$ and thus does not contain x in particular. Since x is a true point, necessarily $i \in \{2, \ldots, k-1\}$.

   (b) $p_x = p_{b_i}$:
   It is easy to see that $p_x \in [p_{a_i}, p_{b_i}]$ (otherwise $x \notin [a_i, b_i]$). Since $p_{b_i} = p_{a_i} + 1$ by Item 1a and Observation 5.2.2, p_x is equal to either p_{a_i} or p_{b_i}. It cannot be p_{a_i} since $p_{a_i}s_x = p_{a_i}0^{\{n-l\}} = p_{a_i}0^{\{n-l-1\}}1 - 1 = a_i - 1$ is a false point by Observation 5.2.7, so necessarily $p_x = p_{b_i}$.

2. If $p_x = p_{b_i}$ for some $i \in \{2, \ldots, k-1\}$, then $f_l^n(x) = 1$:

   Let $p_x = p_{b_i}$ for some $i \in \{2, \ldots, k-1\}$ ($x = p_{b_i}0^{\{n-l\}}$). Since $p_{b_i} = p_{a_i} + 1$ by Observation 5.2.2, $x \geq a_i$. Since $s_x = 0^{\{n-l\}}$ and $p_x = p_{b_i}$, $x \leq b_i$. We conclude that $x \in [p_{a_i}, p_{b_i}]$ and thus $f_l^n(x) = 1$.

We conclude that the true points that end with $0^{\{n-l\}}$ are exactly the points $p_{b_i}0^{\{n-l\}}$ for $i \in \{2, \ldots, k-1\}$. □

5.2.4 Interval decomposition spanning set size

Let $f_l^n = f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$ be a k-interval function defined in Definition 5.2.1. In this section we will express the size of the spanning set returned by the interval decomposition algorithm when run on this function. We will denote its size with $id(f_l^n)$.

The interval decomposition algorithm spans each of the intervals separately using the optimization algorithm from Chapter 2, so $id(f_l^n) = \sum_{i=1}^{k} |\mathcal{T}_i|$, where $\mathcal{T}_i$ is the spanning set of the interval $[a_i, b_i]$ returned by the optimization algorithm for 1-interval functions.

For every i, we shall analyze how the 1-interval algorithm solves the instance $[a_i, b_i]$. We will divide the analysis between the intervals for which $p_{b_i} = p_{a_i}$ (that is $i \in \{1, k\}$) and the ones for which $p_{b_i} = p_{a_i} + 1$ (that is $i \in \{2, \ldots, k-1\}$; see Definition 5.2.1 and Observation 5.2.2).

$i \in \{1, k\}$

In case $i \in \{1, k\}$, the endpoints have common prefixes of length l ($p_{b_i} = p_{a_i}$). Thus the instance $[a_i, b_i]$ is reduced to the instance $[s_a, s_b] = [0^{\{n-l-1\}}1, 1^{\{n-l-1\}}0]$.

Recall that $n \geq l + 2$ by Definition 5.2.1.

If $n = l + 2$, the instance $[0^{\{n-l-1\}}1, 1^{\{n-l-1\}}0] = [01, 10]$ is spanned using the procedure shown in Section 2.3.1. This procedure spans the two trivial sub-intervals $[01, 01]$ and $[10, 10]$ separately, producing $2 = n - l$ ternary vectors.

If $n \geq l + 3$, the instance $[0^{\{n-l-1\}}1, 1^{\{n-l-1\}}0]$ is spanned using the procedure shown in Section 2.3.4. This procedure reduces the instance to the instance $[01, 10]$, which again contributes 2 ternary vectors, and adds $n - l - 2$ extra ternary vectors that span the sub-interval $[0^{\{n-l-2\}}10, 1^{\{n-l-2\}}01]$. The spanning set of the interval in this case has the size $2 + n - l - 2 = n - l$.

We conclude that in case $i \in \{1, k\}$, $|\mathcal{T}_i| = n - l$.

$i \in \{2, \ldots, k - 1\}$

In case $i \in \{2, \ldots, k - 1\}$, recall that $p_{b_i} = p_{a_i} + 1$ by Observation 5.2.2. This means that $p_{a_i} = c01^{\{m\}}$ and $p_{b_i} = c10^{\{m\}}$ for some binary vector c and a number $m \in \{0, \ldots, l - 1\}$.

To see that $m \geq 1$, recall that p_{b_i} is even by Observation 5.2.8.

This means that a_i and b_i start with a common prefix c followed by the bits 01 in a_i and 10 in b_i. In the 1-interval optimization algorithm, the common prefix c is left out and then the procedure shown in Section 2.3.1 is applied to the remainder of the endpoints (that is $[011^{\{m-1\}}s_a, 100^{\{m-1\}}s_b]$). This procedure spans the two sub-intervals $[1^{\{m\}}s_a, 1^{\{m+n-l\}}]$ and $[0^{\{m+n-l\}}, 0^{\{m\}}s_b]$ separately using the suffix and prefix procedure respectively (see Section 2.2).

Let us consider the sub-interval $[0^{\{m+n-l\}}, 0^{\{m\}}s_b]$; the other one is symmetric.

Note that this sub-interval can be immediately reduced to $[0^{\{n-l\}}, s_b]$ by removing the common prefix of the endpoints.

Recall that the prefix algorithm produces one ternary vector for each bit that is set to 1 in $s_b + 1 = 1^{\{n-l\}}$. The returned spanning set of $[0^{\{n-l\}}, s_b]$ thus has the size $n - l$. Prepending the prefix p_{b_i} we get a spanning set of $[p_{b_i}0^{\{n-l\}}, p_{b_i}s_b]$.

We span the complementary sub-interval $[p_{a_i}s_a, p_{a_i}1^{\{n-l\}}]$ in a symmetric manner, getting another spanning set of size $n - l$.

Joining the sub-interval spanning sets together we get the spanning set $\mathcal{T}_i$ of size $2(n - l)$.

From the above considerations we can conclude the following proposition:

Lemma 5.2.1. $id(f_l^n) = 2^l(n - l)$

Proof.

$$id(f_l^n) = \sum_{i=1}^{k} |\mathcal{T}_i| = 2(n-l) + \sum_{i=2}^{k-1} 2(n-l) = 2(n-l) + (k-2)2(n-l)$$
$$= 2(n-l)(k-1) = 2(n-l)((2^{l-1}+1)-1) = 2^l(n-l)$$

□

5.2.5 Minimum spanning set size

In this section we will show a spanning set of f_l^n of size $n+l-2$ and prove its optimality ($dnf(f_l^n) = n+l-2$).

Lemma 5.2.2. $dnf(f_l^n) \leq n+l-2$

Proof. We will show a spanning set $\mathcal{T}$ of f_l^n of size $n+l-2$. We will span the union of the intervals $[p0^{\{n-l-1\}}1, p1^{\{n-l-1\}}0]$ for all $p \in \{0,1\}^l$ using $n-l$ ternary vectors and then we will add $2l-2$ vectors to span the remaining true points.

First recall that by Observation 5.2.10 all the false points of f_l^n end with either $0^{\{n-l\}}$ or $1^{\{n-l\}}$. This means we need to span all the points that end with different $(n-l)$-bit suffixes. We can do that with the set $\mathcal{T}_{out}$ defined in the following way:

$$\begin{aligned}\mathcal{T}_{out} &= \{\phi^{\{l\}}s | s \text{ is a cyclic shift of } 01\phi^{\{n-l-2\}}\} \\ &= \{\phi^{\{l\}}01\phi^{\{n-l-2\}}, \phi^{\{l\}}\phi 01\phi^{\{n-l-3\}}, \ldots, \phi^{\{l\}}1\phi^{\{n-l-2\}}0\}\end{aligned}$$

To see that $\mathcal{T}_{out}$ spans all the points that do not end with $0^{\{n-l\}}$ or $1^{\{n-l\}}$, note that for every $(n-l)$-bit binary vector x that contains both a 0 and a 1 as its component, there is a position i such that $x^{[i]} = 0$ and $x^{[(i+1) \bmod (n-l)]} = 1$. Thus we have spanned the union of the intervals $[p0^{\{n-l-1\}}1, p1^{\{n-l-1\}}0]$ for all $p \in \{0,1\}^l$ using $n-l$ ternary vectors.

There are still some true points left for us to span, namely those that end with $0^{\{n-l\}}$ or $1^{\{n-l\}}$. Recall that by Observation 5.2.11 the true points that end with $0^{\{n-l\}}$ are exactly all the points $p_{b_i}0^{\{n-l\}}$ for $i \in \{2, \ldots, k-1\}$. Also recall that all such p_{b_i} are even numbers by Observation 5.2.8. In fact, p_{b_i} for $i \in \{2, \ldots, k-1\}$ are all the l-bit even numbers except $0^{\{l\}}$ (this is easy to see from Observation 5.2.9 and the fact that $p_{b_1} = 0^{\{l\}} < p_{b_2}$). We will employ the technique of cyclic shifts again to span these points efficiently.

Let $\mathcal{T}_0$ be the set defined as follows:

$$\mathcal{T}_0 = \{c00^{\{n-l\}} | c \text{ is a cyclic shift of } 1\phi^{\{l-2\}}\}$$

To see that $\mathcal{T}_0$ spans all the true points that end with $0^{\{n-l\}}$, note that the cyclic shifts of $1\phi^{\{l-2\}}$ span exactly all the $(l-1)$-bit numbers that contain a 1 and the vectors $c0$ where c is a cyclic shift of $1\phi^{\{l-2\}}$ span exactly all the l-bit *even* numbers that contain a 1. Clearly $|\mathcal{T}_0| = l-1$.

Symetrically we construct $\mathcal{T}_1$ of size $l-1$ that spans all the true points that end with $1^{\{n-l\}}$.

Putting the partial spanning sets together, we get the size of a spanning set of f_l^n:

$$\begin{aligned}|\mathcal{T}| &= |\mathcal{T}_{out}| + |\mathcal{T}_0| + |\mathcal{T}_1| \\ &= (n-l) + (l-1) + (l-1) \\ &= n+l-2\end{aligned}$$

□

Having shown a spanning set of f_l^n of size $n+l-2$, we conclude that $dnf(f_l^n) \leq n+l-2$.

By further analysis we can even show that $dnf(f_l^n) = n+l-2$. Although this fact is not needed to show the lower bound of the approximation ratio, we include it here for completeness.

Lemma 5.2.3. $dnf(f_l^n) \geq n+l-2$

Proof. We will show that f_l^n can not be spanned using fewer than $n+l-2$ ternary vectors by showing an orthogonal set of f_l^n of size $n+l-2$. By Theorem 1.6.1, this proves that all the spanning sets of f_l^n have the size at least $n-l-2$.

The orthogonal set V consists of 3 components:

- $V_{out} = \{0^{\{l\}}s | s \text{ is a cyclic shift of } 10^{\{n-l-1\}}\}$
- $V_0 = \{c00^{\{n-l\}} | c \text{ is a cyclic shift of } 10^{\{l-2\}}\}$
- $V_1 = \{c11^{\{n-l\}} | c \text{ is a cyclic shift of } 01^{\{l-2\}}\}$

To see that the sets V_{out}, V_0 and V_1 are pairwise disjoint, consider any $v_{out} \in V_{out}$, $v_0 \in V_0$ and $v_1 \in V_1$. The vector v_1 differs from both v_{out} and v_0 in the l-th position. The vector v_0 differs from v_{out} in the position $i \in \{1, \ldots, l-1\}$ such that $v_0^{[i]} = 1$.

Since the sets V_{out}, V_0 and V_1 are pairwise disjoint, their union V has the size $|V_{out}| + |V_0| + |V_1| = (n-l) + (l-1) + (l-1) = n+l-2$.

For V to meet the criteria of orthogonality by Definition 1.6.1, we need to show that every vector in V is a true point of f_l^n and that these vectors are pairwise orthogonal.

Claim 5.2.2. $V \subseteq TP(f_l^n)$

Proof. Let $v \in V$. Let $p_v = v^{[1,l]}$ and $s_v = v^{[l+1,n]}$. We need to show that v is a true point of f_l^n. We will do this separately for each of the subsets V_{out}, V_0 and V_1 of V.

1. $v \in V_{out}$:

   Let $v = 0^{\{l\}}s \in V_{out}$. Since s contains a 1, it is different from $0^{\{n-l\}}$. Thus it is not a false point by Observation 5.2.10.

2. $v \in V_0$:

   Let $v = c00^{\{n-l\}} \in V_0$. Since $c0$ is an l-bit even number, by Observation 5.2.9 there is $i \in \{1, \ldots, k-1\}$ such that $c0 = p_{b_i}$. Since $c0$ contains a 1 in some position, necessarily $p_{b_i} \neq p_{b_1}$ and thus $i > 1$. It follows from Observation 5.2.11 that $v = p_{b_i}0^{\{n-l\}}$ is a true point.

3. $v \in V_1$:

   This case is symmetric to Case 2.

□

Claim 5.2.3. *Let $x, y \in V$, $x \neq y$. Then x and y are orthogonal with respect to f_l^n.*

Proof. Let $x, y \in V$ and $x \neq y$. Let T span both x and y. We shall show that T necessarily spans a false point of f_l^n. We shall consider the six possible combinations separately:

- $x, y \in V_{out}$: T spans the false point $0^{\{n\}}$.
- $x, y \in V_0$: T spans the false point $0^{\{n\}}$.
- $x, y \in V_1$: T spans the false point $1^{\{n\}}$.
- $x \in V_{out}, y \in V_0$: T spans the false point $0^{\{n\}}$.
- $x \in V_{out}, y \in V_1$: T spans the false point $0^{\{l\}}1^{\{n-l\}}$.
- $x \in V_0, y \in V_1$: T spans the false point $y^{[1,l]}x^{[l+1,n]} = y^{[1,l-1]}10^{\{n-l\}}$. To see that it is a false point, recall that all the true points that end with $0^{\{n-l\}}$ have an *even* prefix of length l by Observations 5.2.8 and 5.2.11.

□

Thus we have found an orthogonal set V of f_l^n of size $n + l - 2$. We conclude that $dnf(f_l^n) \geq n + l - 2$. □

Having proven that $n + l - 2$ is both an upper bound and a lower bound of $dnf(f_l^n)$, we conclude:

Corollary 5.2.4. $dnf(f_l^n) = n + l - 2$

5.2.6 A lower bound of the approximation ratio

In Corollary 5.2.4 we have shown that $dnf(f_l^n) = n + l - 2$. We can use this information to prove that the approximation ratio of the interval decomposition algorithm is close to $2k$ for large k.

The approximation ratio of the interval decomposition algorithm on functions f_l^n for $l \geq 1$ and $n \geq l + 2$, denoted $approx_{ID}(f_l^n)$, can be computed as follows:

$$approx_{ID}(f_l^n) = \frac{id(f_l^n)}{dnf(f_l^n)} = \frac{2^l(n-l)}{n+l-2}$$

We shall analyze the approximation ratio on each of the classes of functions f_l^n with common l. Since all the functions in such class share the number of intervals, we will continue to generalize the result for classes $INT(k)$ for k in form $2^{l-1} + 1$ for $l \geq 1$.

Definition 5.2.2. The class $DIFFICULT(l)$ contains exactly all the functions f_l^n for $n \geq l + 2$:

$$DIFFICULT(l) = \{f_l^n | n \geq l + 2\}$$

Observation 5.2.12. *Let $l \geq 1$. Then $approx_{ID}(DIFFICULT(l)) = 2^l$.*

Proof.

$$\begin{aligned} approx_{ID}(DIFFICULT(l)) &= \sup_{f \in DIFFICULT(l)} \frac{id(f)}{dnf(f)} \\ &= \sup_{n \geq l+2} \frac{id(f_l^n)}{dnf(f_l^n)} \\ &= \sup_{n \geq l+2} \frac{2^l(n-l)}{n+l-2} \\ &= 2^l \end{aligned}$$

To see that for any $l \geq 1$ the number 2^l really is the supremum of the given sequence, note that $\frac{n-l}{n+l-2} = 1 - \frac{2l-2}{n+l-2} \leq 1$ for all $n \geq l+2$ and that $\lim_{n\to\infty} \frac{2^l(n-l)}{n+l-2} = \lim_{n\to\infty} \frac{2^l n}{n} = 2^l$. □

In other words, for every integer $l \geq 1$ and any real $\varepsilon > 0$ there is some $n \geq l+2$ such that $approx_{ID}(f_l^n) > 2^l - \varepsilon$.

Having derived the approximation ratio of the interval decomposition algorithm for the class $DIFFICULT(l)$ for every $l \geq 1$, we proceed to show what this means in terms of the number of intervals k.

Observation 5.2.13. *Let k be any number that is equal to $2^{l-1}+1$ for some $l \geq 1$. Then $approx_{ID}(INT(k)) \geq 2k-2$.*

Proof. Let k and $l \geq 1$ be numbers such that $k = 2^{l-1}+1$. It follows from definition of f_l^n that $DIFFICULT(l) \subseteq INT(2^{l-1}+1) = INT(k)$, an approximation ratio on $DIFFICULT(l)$ gives a lower bound of the approximation ratio on the whole $INT(k)$. We know from Observation 5.2.12 that $approx_{ID}(DIFFICULT(l))$ equals 2^l and since $l = \log_2(k-1)+1$ we get that $approx_{ID}(DIFFICULT(l)) = 2(k-1) = 2k-2$. We conclude:

$$approx_{ID}(INT(k)) \geq approx_{ID}(DIFFICULT(\log_2(k-1)+1)) = 2k-2$$

□

Corollary 5.2.5. $\limsup_{k\to\infty} \frac{approx_{ID}(INT(k))}{2k} = 1$

Proof. From Observation 5.2.1 we see that $approx_{ID}(INT(k)) \leq 2k$ for every $k \geq 1$, so $\frac{approx_{ID}(INT(k))}{2k} \leq 1$.

From Observation 5.2.13 we know that we can find arbitrarily large k such that $approx_{ID}(INT(k)) \geq 2k-2$.

$$\limsup_{k\to\infty} \frac{approx_{ID}(INT(k))}{2k} \geq \lim_{k\to\infty} \frac{2k-2}{2k} = 1$$

We conclude:

$$\limsup_{k\to\infty} \frac{approx_{ID}(INT(k))}{2k} = 1$$

□

In other words, for any real number $\varepsilon > 0$ we can find an integer $k \geq 1$ and a k-interval function f such that $approx_{ID}(f) > (2-\varepsilon)k$.

5.2.7 Discussion

To demonstrate the results of this chapter, Table 5.1 shows the bounds on the values of $approx_{ID}(INT(k))$ derived in this chapter for some small values of k such that $k = 1$ or $k = 2^{l-1} + 1$ for some l.

k	Lower bound	Upper bound
1	1	1
2	2	2
3	4	6
5	8	10
9	16	18

Table 5.1: Bounds of $approx_{ID}(INT(k))$ for some small values of k

We shall conclude the chapter with an example of an application of the results.

Example 5.1. Let us suppose our colleague believes that the interval decomposition algorithm is k-approximation. We can use the results shown in this chapter to provide them with a counterexample, that is a function on which the interval decomposition algorithm performs strictly worse than k-approximation.

In Table 5.1 we can see that the algorithm actually is k-approximation for $k \in \{1, 2\}$. However, for k as small as 3, the algorithm is 4-approximation at best. This means that for any real number $r < 4$ we can find a 3-interval function on which the interval decomposition algorithm performs at least r times worse than the optimum. The counterexample function that we will construct will be f_l^n for some values l and n according to the definition in Section 5.2.3.

Our choice of $k = 3$ restricts l to the value 2. Choosing the value of r very close to 4 would lead to the construction of a function with large arity n. Since we would rather present a simple example, let us simply require r to be strictly larger than 3 and calculate a sufficiently large n.

We need to find the value of n such that $id(f_2^n) > 3 \cdot dnf(f_2^n)$. Expanding the inequality by using Lemma 5.2.1 and Corollary 5.2.4, we get a restriction on n:

$$\begin{aligned} id(f_l^n) &= 2^l(n-l) = 2^2(n-2) = 4(n-2) \\ &= 4n - 8 > 3n \\ &= 3 \cdot (n + 2 - 2) = 3 \cdot (n + l - 2) = 3 \cdot dnf(f_l^n) \end{aligned} \tag{5.4}$$

Any $n > 8$ satisifes inequality (5.4):

$$\begin{aligned} 4n - 8 &> 3n \\ n - 8 &> 0 \\ n &> 8 \end{aligned}$$

Let us choose $n = 9$ to keep the example as simple as possible.

We conclude that the 3-interval function f_2^9 can be spanned using $dnf(f_2^9) = 9$ ternary vectors, while the interval decomposition algorithm uses $id(f_l^n) = 28$ vectors, which is strictly more than $k \cdot dnf(f_2^9) = 27$. Moreover, we can construct a spanning set of f_2^9 of size 9 using the technique shown in the proof of Lemma 5.2.2.

Conclusion

The central question of this thesis is finding minimum spanning sets of k-interval Boolean functions for general k.

Schieber et al. showed a polynomial algorithm that minimizes spanning sets of 1-interval functions [8, Section 3]. Schieber et al.'s proof of optimality of the algorithm is based on the fact that every 1-interval function is coverable.

While Dubovský showed a polynomial minimization algorithm for the class of 2-switch 2-interval functions [4, Section 4], we presented a simpler algorithm solving the same problem. The algorithm presented in Section 3.1 reduces every 2-switch 2-interval function to a 1-interval function. The proofs of optimality of both of these algorithms are based on the fact that every 2-switch 2-interval function is coverable.

Dubovský showed a 3-switch 2-interval function that is not coverable [4, p. 32]. We extended his result in Section 3.2 by showing a non-coverable l-switch function for every $l \geq 3$. These functions include some k-interval function for every $k \geq 2$. This suggests that a possible future algorithm that optimally spans a whole class of k-interval functions for any $k \geq 2$ is going to require a different approach for proving optimality.

In Chapter 4 we presented a polynomial-time $2k$-approximation algorithm for minimizing spanning sets of k-interval functions for general k. The algorithm naturally extends Schieber et al.'s suffix-prefix approximation algorithm for disjoint spanning sets of 1-interval functions [8, Section 6].

In Chapter 5 we proposed an improved approximation algorithm for k-interval functions. This algorithm has the approximation ratio 1 on 1-interval functions and the approximation ratio 2 on 2-interval functions. We continued to show an upper bound $2k$ and a lower bound $2k - 2$ on the approximation ratio for every k that is of the form $2^{l-1} + 1$ for some l. These bounds are especially interesting because their ratio converges to 1 for large k.

We mentioned in Section 2.4 that Schieber et al. also introduced a polynomial algorithm that finds a minimum disjoint spanning set representing a 1-interval function [8, Section 4]. Disjoint spanning sets can be especially useful in automatic generating of test patterns [8]. It is worth noting that the $2k$ approximation algorithm we described in Chapter 4 produces a disjoint spanning set for any k-interval function.

An interesting question left to be answered is whether there is a polynomial minimization algorithm for any class of all k-interval functions for $k \geq 2$. Such algorithm would require an innovative approach to proving optimality since all the proofs of optimality mentioned in this thesis depend on coverability of the spanned functions. An approximation algorithm with an approximation ratio lower than $2k - 2$ for all $k \geq 2$ would also be an interesting improvement.

Bibliography

[1] Endre Boros, Ondřej Čepek, Alexander Kogan, and Petr Kučera. Exclusive and essential sets of implicates of Boolean functions. *Discrete Applied Mathematics*, 158(2):81 – 96, 2010. ISSN 0166-218X. doi: http://dx.doi.org/10.1016/j.dam.2009.08.012. Available from: `http://www.sciencedirect.com/science/article/pii/S0166218X09003394`.

[2] Yves Crama and Peter L. Hammer. *Boolean Functions: Theory, Algorithms, and Applications*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2011. ISBN 9781139498630. Available from: `http://books.google.cz/books?id=3KmyKpw_pbUC`.

[3] Richard A. DeMillo and A. Jefferson Offutt. Constraint-based automatic test data generation. *Software Engineering, IEEE Transactions on*, 17(9):900–910, Sep 1991. doi: 10.1109/32.92910. Available from: `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=92910`.

[4] Jakub Dubovský. A construction of minimum DNF representations of 2-interval functions. Master's thesis, Charles University in Prague, 2012. Available from: `https://is.cuni.cz/webapps/zzp/detail/116613/`.

[5] Radek Hušek. Properties of interval Boolean functions. Master's thesis, Charles University in Prague, 2014. Available from: `https://is.cuni.cz/webapps/zzp/detail/145114/`. [In Czech].

[6] Radek Hušek. personal communication, April 2015.

[7] Daniel Lewin, Laurent Fournier, Moshe Levinger, Evgeny Roytman, and Gil Shurek. Constraint satisfaction for test program generation. In *Computers and Communications, 1995., Conference Proceedings of the 1995 IEEE Fourteenth Annual International Phoenix Conference on*. IEEE, Mar 1995, pages 45–48. doi: 10.1109/PCCC.1995.472513. Available from: `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=472513`.

[8] Baruch Schieber, Daniel Geist, and Ayal Zaks. Computing the minimum DNF representation of Boolean functions defined by intervals. *Discrete Applied Mathematics*, 149(1–3):154 – 173, 2005. ISSN 0166-218X. doi: http://dx.doi.org/10.1016/j.dam.2004.08.009. Available from: `http://www.sciencedirect.com/science/article/pii/S0166218X05000752`.

[9] Christopher Umans. The minimum equivalent DNF problem and shortest implicants. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*. IEEE, Nov 1998, pages 556–563. doi: 10.1109/SFCS.1998.743506. Available from: `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=743506`.

[10] Ingo Wegener. *The Complexity of Boolean Functions*. Stuttgart: John Wiley & Sons, and B. G. Teubner, 1987. ISBN 3-519-02107-2. Available from: `http://ecccc.hpi-web.de/static/books/The_Complexity_of_Boolean_Functions/`.

[11] Ondřej Čepek, Petr Kučera, and Petr Savický. Boolean functions with a simple certificate for CNF complexity. *Discrete Applied Mathematics*, 160(4–5): 365 – 382, 2012. ISSN 0166-218X. doi: http://dx.doi.org/10.1016/j.dam.2011.05.013. Available from: `http://www.sciencedirect.com/science/article/pii/S0166218X11001958`.

List of Tables

List of Abbreviations

CNF conjunctive normal form. 9

DNF disjunctive normal form. ii, 1, 3, 4, 6–9