ZDD による
ルールリストポリシーの等価判定

原田 崇司 [1]　田中 賢 [1]　三河 賢治 [2]

[1] 神奈川大学大学院 理学研究科 理学専攻 情報科学領域

[2] 新潟大学学術情報機構情報基盤センター

2018 年 1 月 30 日

AL・FPAI, 大阪府立大学 I-site なんば

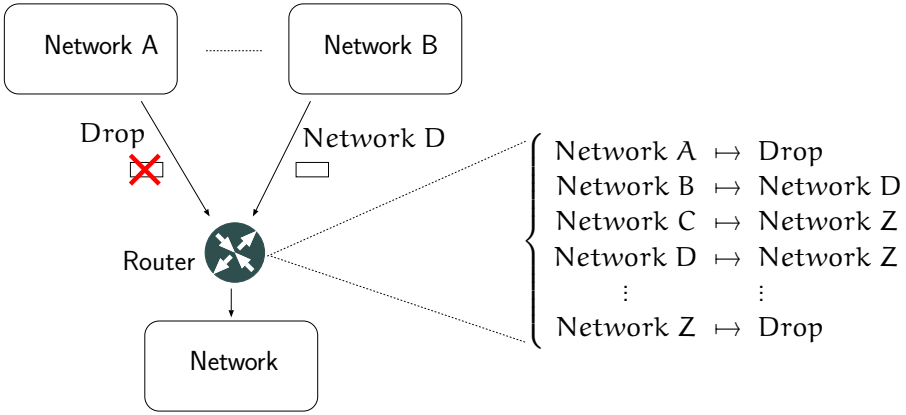
目次

ネットワーク脅威・ネットワーク運用

- 不正アクセス，情報漏洩，DDoS 攻撃，...
- QoS，負荷分散，...

パケット分類

パケット分類



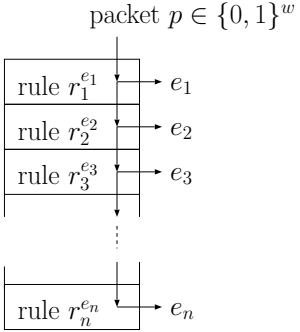
ポリシに従ってパケットを分類

パケット分類

ポリシーを満たすルールリストを作成

このルールリストによってパケットを分類

($r_1, r_2, \ldots, r_{n-1}$ の順でパケットと照合，
最初に合致したルールのアクションを適用)



$r_1 \ldots r_{n-1}$ に合致しないパケットは無条件に e^n を適用

研究背景

ルール数が増加するとパケット分類による遅延が発生

この遅延を減らしたい

到着頻度の高いパケットに合致するルールを上位に配置すると遅延が減少

研究背景

種々のルールリスト再構築・ルール並び替え法

生成されたルールリストが元のポリシーを満たすか確認していない

ポリシーチェッカが必要

関連研究

ネットワーク全体に関する設定，到達可能性・ネットワークループ，...

- L. Yuan, et al., “FIREMAN: a toolkit for firewall modeling and analysis,” S & P, 2006
- P. Kazemian et al., “Real time network policy checking using header space analysis,” NSDI, 2013
- T. Inoue et al., “An efficient framework for data–pane verification with geometric windowing queries,” IEEE Network and Service Management, 2017

etc.

研究目的

二つの決定リストの等価判定問題は **coNP** 完全

⇨ ルールリストポリシーの等価判定も **coNP** 完全

- ZDD は疎な組合せ集合を効率よく表現
- アクションが二つのルールリストポリシーは ZDD で表現可能
- ルールリストのポリシーの組合せ集合は疎

1. ルールリスト $\mathcal{R}_1$ と $\mathcal{R}_2$ に対応する ZDD Z_1 と Z_2 を構築
2. Z_1 と Z_2 が等価か判定（Z_1 と Z_2 のアドレスが同じかを比較）

ルールの形式

一般にパケット分類には，パケットヘッダの

送信元アドレス	(e.g. 131.10.42.40)
宛先アドレス	(e.g. 95.184.130.35)
送信元ポート番号	(e.g. 2020)
宛先ポート番号	(e.g. 22)
プロトコル	(e.g. TCP)

を使用するので，パケット分類のルールは，これら 5 つの項目を指定

(e.g. r_1 : 131.10.42.40/32, 95.184.130.35/32, 0 : 65535, 1724 : 1724, UDP)

抽象化してパケット（ヘッダ）を $0, 1$ の系列，ルールを $0, 1, *$ の系列とみる

ルールの形式

- ルール $r_i^e = b_1 b_2 \dots b_w$ は，
  - ルール番号 $i \in \mathbb{N}$
  - 条件 $b_1 b_2 \dots b_w \in \{0, 1, *\}^w$
  - アクション $e \in \{A_1, A_2, \dots, A_m\}$

  の三項組．ただし，w は条件の長さ，
  '$*$' は don't care，m はアクションの数
  右の例では，$m = 2$ で，$A_1 = P, A_2 = D$

- パケットは長さ w のビット列

Filter $\mathcal{R}$
$r_1^P = 0 * 1 *$
$r_2^D = 0\ 0\ 0\ 0$
$r_3^D = * 0\ 0 *$
$r_4^P = * 1 * 0$
$r_5^P = 1 * 1 *$
$r_6^P = * * 1 *$
$r_7^D = * * * *$

e.g. パケット 1111 は，$r_1^P \dots r_4^P$ に合致せず r_5^P に合致

$\Longrightarrow$ r_5^P のアクション P を適用

ルールリストのポリシー

表 1: ルールリスト

Filter $\mathcal{R}$
$r_1^D = * 1 0 0$
$r_2^P = 0 1 * *$
$r_3^D = * 0 0 1$
$r_4^P = * * * 1$
$r_5^P = 0 * 1 *$
$r_6^D = * 1 1 0$
$r_7^P = * 1 * *$
$r_8^D = * * * *$

表 2: 表 1 が表す函数

$0000 \mapsto D$	$1000 \mapsto D$
$0001 \mapsto D$	$1001 \mapsto D$
$0010 \mapsto P$	$1010 \mapsto D$
$0011 \mapsto P$	$1011 \mapsto P$
$0100 \mapsto D$	$1100 \mapsto D$
$0101 \mapsto P$	$1101 \mapsto P$
$0110 \mapsto P$	$1110 \mapsto D$
$0111 \mapsto P$	$1111 \mapsto P$

ルールリストのポリシーとは，パケットの集合 $\{0,1\}^w$ から
アクションの集合 $\{A_1, A_2, \ldots, A_m\}$ への函数

ルールリストのポリシー

表 1: ルールリスト

Filter $\mathcal{R}$
$r_1^D = * 1 0 0$
$r_2^P = 0 1 * *$
$r_3^D = * 0 0 1$
$r_4^P = * * * 1$
$r_5^P = 0 * 1 *$
$r_6^D = * 1 1 0$
$r_7^P = * 1 * *$
$r_8^D = * * * *$

表 2: 表 1 が表す函数

$0000 \mapsto D$	$1000 \mapsto D$
$0001 \mapsto D$	$1001 \mapsto D$
$0010 \mapsto P$	$1010 \mapsto D$
$0011 \mapsto P$	$1011 \mapsto P$
$0100 \mapsto D$	$1100 \mapsto D$
$0101 \mapsto P$	$1101 \mapsto P$
$0110 \mapsto P$	$1110 \mapsto D$
$0111 \mapsto P$	$1111 \mapsto P$

ルールリスト $\mathcal{R}$ 自体も函数と見做し，
$\mathcal{R}(p)$ はパケット p に対して $\mathcal{R}$ が与えるアクション
e.g. $\mathcal{R}(0101) = P$, $\mathcal{R}(1110) = D$

ポリシー等価判定問題

入力　ルールリスト $\mathcal{R}_1, \mathcal{R}_2$
問い　$\forall p \in \{0,1\}^w \ \mathcal{R}_1(p) = \mathcal{R}_2(p)$?

表 3: ルールリスト $\mathcal{R}_1$

Filter $\mathcal{R}$
$r_1^D = * 1 0 0$
$r_2^P = 0 1 * *$
$r_3^D = * 0 0 1$
$r_4^P = * * * 1$
$r_5^P = 0 * 1 *$
$r_6^D = * 1 1 0$
$r_7^P = * 1 * *$
$r_8^D = * * * *$

表 4: ルールリスト $\mathcal{R}_2$

Filter $\hat{\mathcal{R}}$
$r_1^P = 1 0 1 0$
$r_2^P = * 0 1 *$
$r_3^P = 0 1 1 0$
$r_4^D = * 1 * 0$
$r_5^P = * 1 * *$
$r_6^D = * * * *$

ポリシー等価判定問題は **coNP** 完全

提案手法

- ZDD は疎な組合せ集合を効率よく表現
- アクションが二つのルールリストポリシーは ZDD で表現可能
- ルールリストのポリシーの組合せ集合は疎

1. ルールリスト $\mathcal{R}_1$ と $\mathcal{R}_2$ に対応する ZDD Z_1 と Z_2 を構築
2. Z_1 と Z_2 が等価か判定（Z_1 と Z_2 のアドレスが同じかを比較）

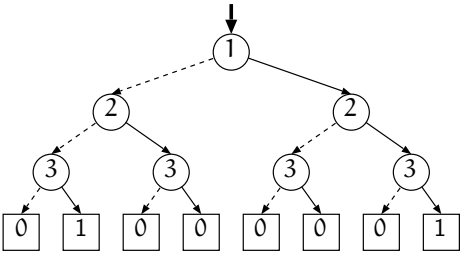
BDD/ZDD

BDD (Binary Decision Diagram)

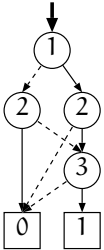
論理函数を効率よく扱えるデータ構造

ZDD (Zero-Suppressed Binary Decision Diagram)

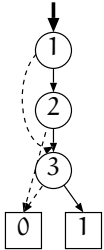
疎な組合せ集合を効率よく扱えるデータ構造



場合分け二分木



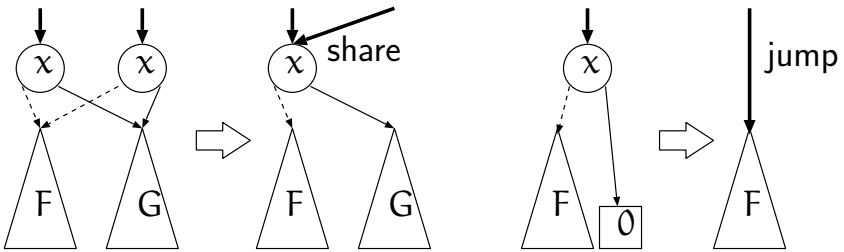
BDD



ZDD

ZDD の節点削除規則

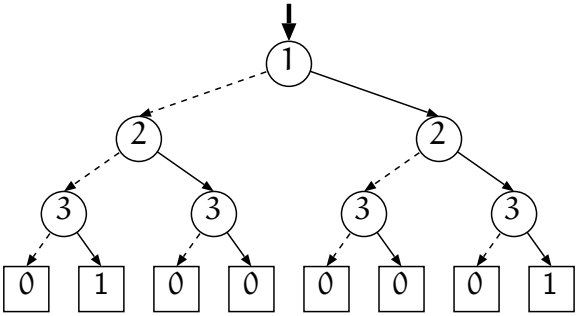
論理函数（組合せ集合）に対する場合分け二分木に対して，以下の節点削除規則を既約になるまで適用すれば ZDD の出来上がり



等価節点の共有　　冗長節点の削除

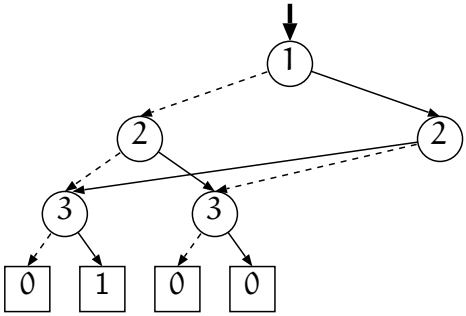
場合分け二分木を構築してから簡約することによって ZDD を構築するのは遅すぎるので，ZDD を高速に構築するための技法が数多存在

場合分け二分木から ZDD 構築の例



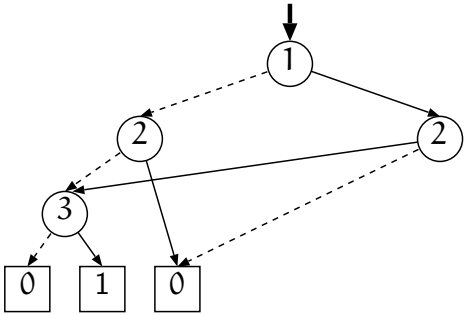
場合分け二分木（初期状態）

場合分け二分木から ZDD 構築の例



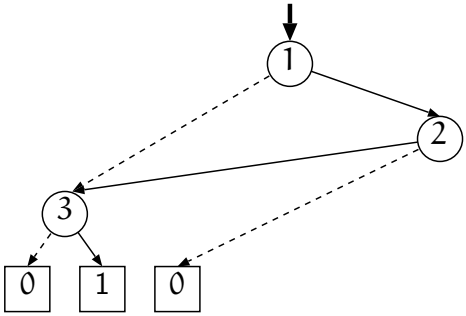
等価節点（0 枝と 1 枝の先が同じ節点 u と v）の共有

場合分け二分木から ZDD 構築の例



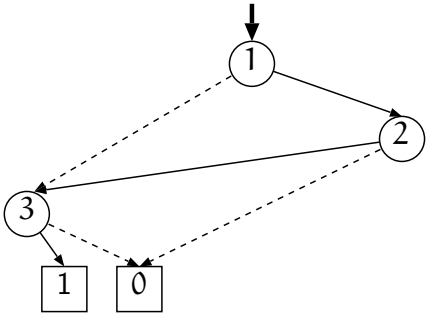
冗長節点（1 枝の先が 0 終端節点を指す節点）の削除

場合分け二分木から ZDD 構築の例



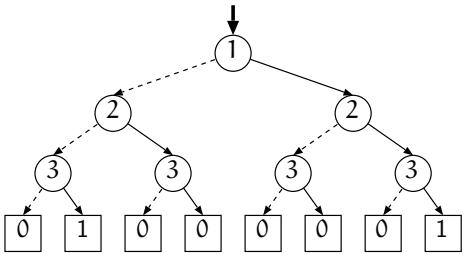
冗長節点（1 枝の先が 0 終端節点を指す節点）の削除

場合分け二分木から ZDD 構築の例

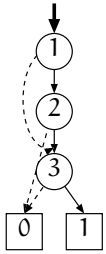


等価節点（0 枝と 1 枝の先が同じ節点 u と v）の共有

ZDD の読み方



場合分け二分木



ZDD

根節点から終端節点 $\boxed{0}$, $\boxed{1}$ に向かって辿る

- 変数 $\textcircled{i}$ から 0 枝 (破線) を辿れば $x_i = 0$，1 枝 (実線) を辿れば $x_i = 1$
- 変数 $\textcircled{i}$ から 0 枝を辿った先が変数 $\textcircled{j}$ ならば，$x_{i+1}, x_{i+2}, \ldots, x_{j-1} = 0$

例えば，$\textcircled{1}$ --→ $\textcircled{3}$ ⟶ $\boxed{1}$ は，$f(x_1 = 0, x_2 = 0, x_3 = 1) = 1$ を意味

複数の ZDD の共有化（共通の部分グラフを共有）

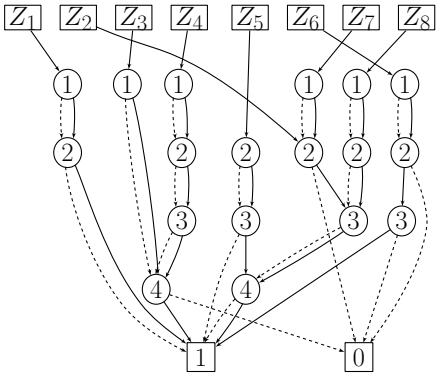


図 1: 複数の ZDD の共有化

$$Z_2 = \{0100, 0101, 0110, 0111\}$$
$$Z_7 = \{0100, 0101, 0110, 0111, 1100, 1101, 1110, 1111\}$$

Z_2 と Z_7 は部分グラフを共有

ルール r_i^e に対する ZDD 構築

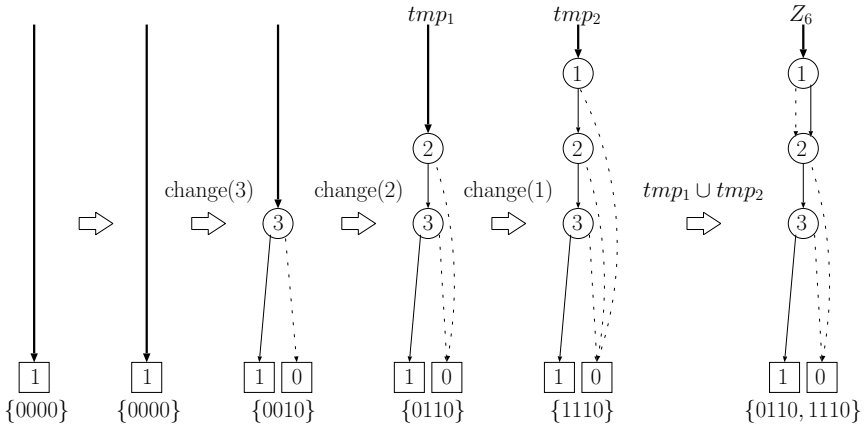


図 2: $r_6^D = *110$ に対応する ZDD の構築過程

ルールリスト $\mathcal{R}$ に対する ZDD 構築

アクション D が適用されるパケットの集合に対応する ZDD を構築

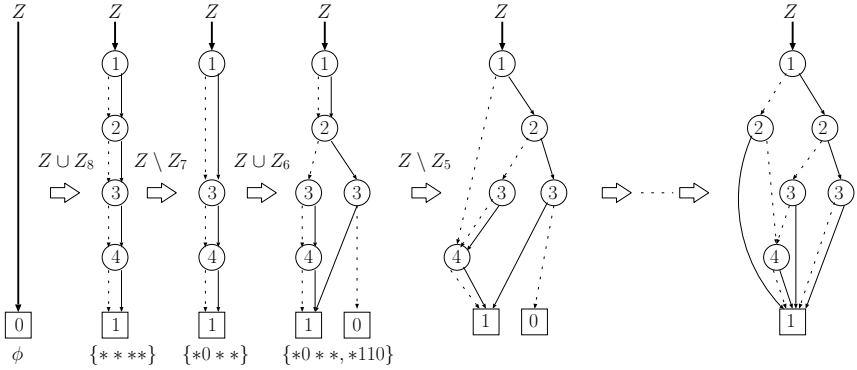


図 3: ルールリスト $\mathcal{R} = \langle \mathrm{r}_1^{\mathrm{D}}, \cdots, \mathrm{r}_5^{\mathrm{P}}, \mathrm{r}_6^{\mathrm{D}}, \mathrm{r}_7^{\mathrm{P}}, \mathrm{r}_8^{\mathrm{D}} \rangle$ に対する ZDD の構築過程

多値のルールリスト $\mathcal{R}_1$ と $\mathcal{R}_2$ のポリシー等価判定

ここまでは，アクションが P と D 二つのルールのみを考慮

アクションが複数 $A_1, A_2, \ldots, A_m$ のルールリスト $\mathcal{R}_1$ と $\mathcal{R}_2$

⇩

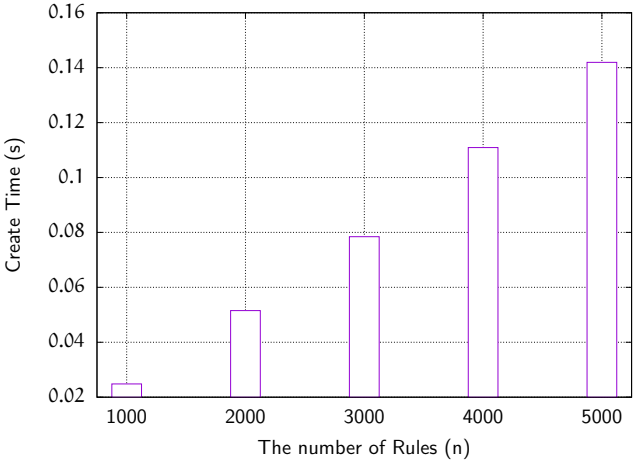
アクションの数 m だけそれぞれ ZDD を構築し，それぞれ等しいか確認する手法を提案

実験環境

OS	:	CentOS Release 6.10 (Final)
CPU	:	Intel Core i5-3470 3.20 GHz
主記憶容量	:	2GB
実装言語	:	C
コンパイラ	:	gcc version 4.4.7

- ルール長 $w = 104$，ルール数 $\mathrm{n} = 1000, 2000, \ldots, 5000$ のルールリストを ClassBench を用いて生成，アクションは P と D の二つを付与
- 判定時間 (s) を計測

実験結果：判定時間（秒）



ルール数 5000 のルールリストに対して 0.2 秒未満で等価判定

まとめと今後の課題

まとめ

ルールリスト再構築・ルール並び替えの後にはポリシーチェックが必要

⇨ ZDD を用いたルールリストの等価判定プログラムを作成

今後の課題

- 多値（アクションが三つ以上）のルールリストに対する提案手法の有効性確認
- ルール数 1 万から 100 万までの巨大なルールリストに対しての有効性確認
- 仮想スイッチやルータのルールに対しての有効性確認