知識ベースに基づくネットワークトラブルシューティングの自動化

紅林 輝[†]，梶 克彦[†]，河口 信夫[†]

[†] 名古屋大学大学院工学研究科

ネットワークの大規模化や構成の複雑化に伴い，ネットワークで生じたトラブルの原因の特定が困難になり，トラブルシューティングはネットワーク管理者の負担を大きくする要因の一つとなる．本研究は，ネットワークのトラブルの原因を自動推定するシステムを開発し，ネットワーク管理者の負担の軽減を目的とする．本研究では，あらかじめネットワークに関する知識や，トラブルの原因を推定するための知識を，知識ベースとしてプログラムに記述しておく．その知識ベースを元に，トラブルの原因を推定するために２段階の推論を行うことで原因を推定する．

Automation of Network Troubleshooting based on Knowledge Base

Akira Kurebayashi[†], Katsuhiko Kaji[†], Nobuo Kawaguchi[†]

[†] Graduate School of Engineering, Nagoya University

Increasing of network size and complexity makes th network troubleshooting more difficult. This research aims to develop a system to automatically estimate the cause of network problems. In this research, we describe network and cause of the trouble by knowledge base. We estimate the cause of the trouble by two-step inference based on the knowledge base.

1　はじめに

ネットワークにトラブルが発生し，ネットワークの利用が長時間困難になると，利用しているユーザに大きな迷惑をかけてしまうため，トラブルシューティングはネットワーク管理者の重要な仕事の一つである．ネットワークの大規模化や構成の複雑化に伴い，トラブルの原因を特定するのは難しくなる．例えば，近年では，VLAN（仮想 LAN）の利用により，多様な論理ネットワークを一つの物理的回線の上に実現可能となっている．ただし，VLAN を適切に利用するためには，VLAN を構成する全てのスイッチに対して適切な設定を行う必要がある．したがって，ネットワークにトラブルが発生した際に，VLAN の設定が正しくされているかを確認するためには，全てのスイッチの Config を確認する必要があり，非常に大きな労力を必要とする．

また，比較的小さな実験用のネットワークを構築する際にも，Config のミスにより正しい構築に時間がかかる場合がある．

本研究は，ネットワークのトラブルの原因を自動推定するシステムを開発し，ネットワーク管理者の負担の軽減を目的とする．

本研究のアプローチとして，人工知能等の分野で研究が進められてきたルールベースの手法を用いる．ネットワークのシステムは正しく動作するための仕様が決まっているため，それらの知識をルールとして網羅すれば，原因の推定が可能であると考えられる．

また，トラブルの原因推定のために，実際のネットワークから接続情報や Config の情報等を取得する必要がある．本研究では，ネットワーク情報の取得を外部のネットワーク管理システムである NGMS[6] を利用し取得することを想定する．したがって，本研究では，得られた情報からいかにして原因を推定するかという点に焦点を当てる．

本研究では，Config 以外の原因にも対応することを想定しているが，今回，VLAN の Config に関係したトラブルの原因を推定するルールの実装と動作のテストを行った．

本稿の構成は以下の通りである．2 章で原因推定の手法とそれを構成する知識ベースの設計と推論の流れを述べる．3 章で，実装したシステムについて述べる．4 章で，関連研究について述べる．最後に 5 章で，まとめと今後の課題について述べる．

2　知識ベースの設計

2.1　Prolog による記述

本研究では，論理型言語である Prolog を用いて，ルールを記述し，知識ベースを構築する．Prolog は 1972 年に Colmerauser らにより開発されたプログラミング言語であり，特に AI プログラムの記述に用いられている．

Prolog は，述語論理に基づいており，事実と規則（ルール）を記述することでプログラムを構築する．

事実の記述に関して，例えば，「Catalyst3750 は L3 スイッチである」という事実は「l3switch(cat3750).」と記述し，「cat3750」が引数となり「l3switch」が述語となる．

小文字で記述した文字列をアトムと呼び，大文字で表された文字列は変数となる．Prolog は引数に変数を記述しゴール節として実行することで，記述された事実に基づいて変数とアトムが一致するものを全て探索する．

例えば,「l3switch(X)」をゴール節とし実行すると，バックトラックによって変数 X に全ての L3 スイッチとして記述されたアトムが代入され，全ての L3 スイッチを求める事ができる．

ルールの記述に関して，Prolog では以下のような

IF A1, A2 THEN B(A1 かつ A2 が成り立てば B)

というルールを次のような定義節で表現する．

B :- A1, A2.

これを利用して，例えば，

can_config_vlan(X) :- l3switch(X).

という推論規則を記述すれば，直接表されていない他の関係，ここでは「もし，X が L3 スイッチであれば，X は VLAN を Config できる」ということが推論によって分かる．

Prolog は，このように記述された事実とルールに基づいて，後ろ向き推論方式に従ってバックトラックを続け，全ての答えを探索する．このように Prolog はルールの記述の簡易さに優れており，さらに，論理的に全ての答えを求めて推論を行う機構は，ネットワークの接続情報や VLAN 情報を扱うのに適していると考えられる．

2.2　扱うトラブルの範囲

ネットワーク管理の範囲は広く多岐にわたるため，本研究では，トラブルの種類を以下のものに限定している．物理的な障害やネットワーク機器の Config 等，基本的なトラブルの原因を含んでおり，以下のトラブルに関して正しく原因を推定出来れば本研究のアプローチは有効だといえる．

- 物理的な障害
  - 機器
  - ポート
  - ケーブル
- IP アドレスの競合
  - ホストとゲートウェイ
  - ホスト同士
  - ネットワーク機器同士
- Config ミス
  - VLAN トランク/アクセスポートのモード違い
  - ポートに当てる VLAN ID 違い
  - スイッチ自体への VLAN 設定違い
  - IP 経路違い
  - ポートの shutdown ミス

現在，これらのうち，VLAN の Config に関する以下の原因についての実装と動作テストを行った．

- VLAN　トランク/アクセスポートのモード違い
- ポートに当てる VLAN ID 違い
- スイッチ自体への VLAN 設定違い

次節では，それらに関する述語の設計及び推論の流れについて述べる．

2.3　原因推定のアプローチ

原因推定までの概要を図 1 に示す．

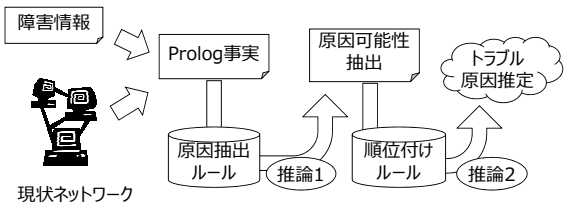

図 1: 原因推定の概要

本システムは，2 段階の推論を行うことによって，原因を推定する．まず，考えられるトラブルの原因の可能性を複数抽出する．次に，抽出されたトラブルの原因から，さらに推論を行いそれらの順位付けを行うことで，最終的に原因を推定する．

1 段階のみの推論だと，原因の可能性の低いものも含めて出してしまうため，原因の可能性となる項目が多く出すぎてしまう．推論を 2 段階にすることで，2 段階目で抽出された複数の原因の関係から，その中から原因を特定するルールや，原因の可能性の程度を判定するルールの記述が可能となる．システム全体の動作イメージ図を図 2 に示す．

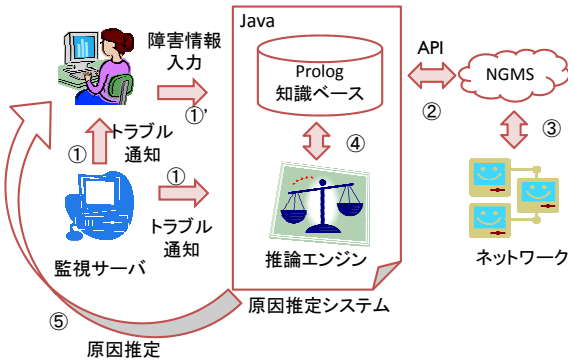

図 2: システム全体の動作イメージ

原因推定までのシステム全体の動作の流れを以下に述べる．まず，障害情報，ネットワークの情報からそれらを表す Prolog の事実を生成する．障害情報はユーザまたは監視サーバと連携して本システムに入力される．その障害情報にしたがってネットワークから情報を取得する．次に，第 1 段階の推論で，原因推定ルールとして記述した Prolog ルールに従い，原因と考えられる項目を抽出する．さらに，抽出された複数の原因可能性から，原因順位付けルールに従い，それぞれの原因可能性の順位付けを行い，原因を推定する．システムの実装については，3 章で述べる．

2.4　具体例

以下に具体例として，「選ばれた複数のスイッチが，ある VLAN でまとめるはずのブロードキャストドメインに入らない」という障害の原因を推定する流れについて述べる．

2.4.1　事実の生成

まず，ユーザあるいはネットワーク監視システムから障害情報が入力され，この障害情報を表す Prolog 事実が生成される．

```
cannot_share_vlan_to_switches(Vlan_id,Switch_list).
```

この述語は，推論の際，次の 2 つの述語を呼び出す．

Switch1 と Switch2 では Vlan_id のブロードキャストドメインを共有出来ない．

```
cannot_switch_vlan_to_switch_switch(Switch1,
  Switch2,Vlan_id).
```

Port1 と Port2 では Vlan_id のブロードキャストドメインを共有出来ない．

```
cannot_share_vlan_to_port_port(Port1,Port2,
  Vlan_id).
```

ここで呼び出されている Switch1 や Port1 という変数は，Switch_list から得られるものである．障害情報から，スイッチや物理ポートを別々にして述語を生成することで，それぞれについての異なるルールを呼び出す事が可能となり，より正確な原因の推定が可能となる．具体的なルールについては後述する．

次に，ネットワークから情報を取得し，得られた情報をもとに，ネットワーク情報の事実が生成される．例えば，現在，cat3750 というスイッチがあり，そのスイッチには Vlan1 が登録されている．そのスイッチには 3 つのポートがあり，その中の 2 番ポートがアクセスポートで Vlan1 が登録されている．以上のような事実であれば以下のように生成される．

```
now_l3sw(cat3750-1).
now_device_have_port_list(cat3750-1,[cat3750
  -1_1,cat3750-1_2,cat3750-1_3]).
now_config_vlan_to_switch(cat3750-1,[1]).
now_config_untag_vlan_id_to_port(cat3750-
  1_2,1).
```

また，事実の生成に関するルールによって，生成済みの事実から新たに事実が生成される．

```
now_device_have_port(Device,Port):-
  now_device_have_port_list(Device,Port_list),
  member(Port,Port_list).
nw_device(Nw_Device) :- l3sw(Nw_Device).
```

2.4.2　原因可能性抽出

次に，障害情報と生成された事実に基づき，Prolog ルールに従って推論を行う．前述したように，原因推定のためのルールとして，トラブルの原因可能性を抽出するルールと，抽出された原因可能性を順位付けするルールが存在する．原因可能性抽出ルールには，トラブルの原因の可能性の有無を判定するための条件が記述されている．例えば以下のようなものである．

[原因可能性抽出ルールの例]

rule1

IF 2 つのスイッチのうち，どちらかに指定の VlanID が登録されていない
THEN その 2 つのスイッチでは VlanID のブロードキャストドメインを共有できない．

rule1 の述語

```
cannot_share_vlan_to_switch_switch(Switch1,_,
  Vlan_id):-
  now_config_vlan_to_switch(Switch1,Vlan_list),
  not(member(Vlan_id,Vlan_list)).
```

rule2

IF 2 つのスイッチが接続されているポートは，トランクポート同士だけれど、どちらかの　ポートに指定の VlanID が登録されてい　ない
THEN その 2 つのスイッチでは VlanID のブロードキャストドメインを共有できない．

rule2 の述語

```
cannot_share_vlan_port_port(Port1, Port2,
  Vlan_id) :-    now_config_tag(Port1),
  now_config_tag(Port2),
  now_config_tag_vlan_id_to_port(Port1,
  Vlan_id_list), not(member(Vlan_id,
  Vlan_id_list)).
```

rule3

IF Port1 と Port2 がトランクポートとアンタグポートで接続されている
THEN Port1 と Port2 では，Vlan_id のブロードキャストドメインを共有出来ない

rule3 の述語

```
cannot_share_vlan_to_port_port(Port1,
  Port2, Vlan_id) :-
  now_config_untag(Port1),
  now_config_tag(Port2),
  my_trouble_assert(trouble_cause_unmatched_
  vlan_mode_to_port_port(Port1, Port2)).
```

これらのルールに従い，条件部に合致したトラブルの原因可能性をシステムに登録していく．2.2 で述べた本稿で対象としている VLAN の Config 間違いについては，上記を含む 8 つのルールで，全てのパターンを網羅しており，原因可能性の抽出が可能となっている．

2.4.3　順位付け

登録された複数の原因可能性から順位付けを行い，原因を推定する．順位付けのルールは例えば以下のようなものである．

[順位付けルール]

rule4

IF 障害情報が複数のスイッチである VLAN のブロードキャストドメインが共有できない，という場合に，スイッチ自体に VLAN の Config が入っていないという原因可能性がある
THEN この原因可能性のランクは 1 である．

rule4 の述語

```
exec_trouble_rank :-
  fail_info_cannot_share_vlan_to_switches(_,_),
  trouble_cause_not_config_vlan_id_to_switch
  (Switch,Vlan_id),my_trouble_rank_assert(1,
  trouble_cause_not_config_vlan_id_to_switch
  (Switch,Vlan_id)).
```

原因ランクは数字が低い方が，トラブルの可能性の順位が高くなる．デフォルトで原因のランクの数値は 100 であり，条件部に合致した原因可能性は，数値を下げてランクを高くする．

3　システム構成

3.1　NGMS

トラブルの原因推定のために，ネットワークから Config，接続状態，トラフィック等の情報を取得する必要がある．本研究では，それらの取得にネットワーク管理システム NGMS[6] を使用することを想定している．NGMS は，名古屋大学を中心として開発が進められているネットワーク管理システムである．ネットワーク運用管理に必要な機能の網羅をコンセプトの 1 つにしており，その中には SNMP によるネットワークの監視や，ネットワーク機器の Config 情報の管理等が含まれる．また，API による情報の提供も構想に含まれており，本研究ではこのシステムを利用し，原因推定に必要な情報を取得することを想定している．

3.2　開発環境

本研究では，開発に Eclipse Rich Client Platform(RCP)[7] を利用している．Eclipse RCP には，Eclipse の提供するワークベンチ，パースペクティブ，エディタ，ビュー等をアプリケーションに組み込めるといったメリットがある．また，ネットワークの描画には，Eclipse のビューや SWT アプリケーションに簡単にグラフィカルツリーの描画機能を組み込めるプラグインである Zest[8] を利用している．

2 章で述べたように，知識ベースの記述と推論には，Prolog を用いている．本研究では，Java 上で動作する Prolog 処理系である tuProlog[2] を利用している．

3.3　実装システム

本研究では，ネットワークの実環境がなくてもルールのテストが出来るよう，ネットワーク機器の Config や障害情報の入力が出来るシステムを開発した．

実装しているシステムの GUI を図 3 に示す．

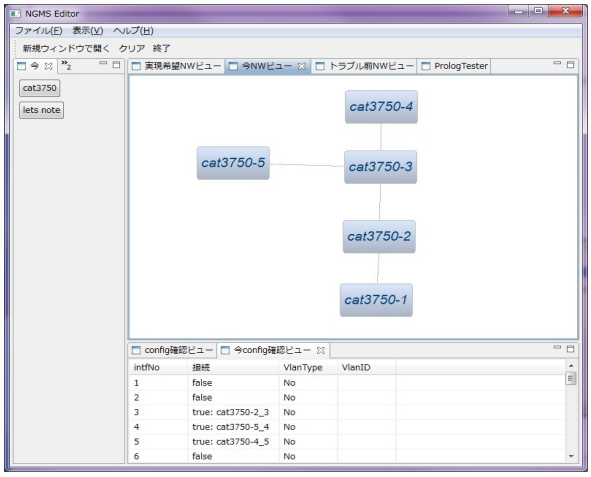

図 3: システム GUI

2.2 で述べたように，現在，VLAN の Config に関するトラブルのルールについての実装と動作テストを行った．

例として，2.3 で述べたケースと同じように「ある VLAN のブロードキャストドメインを共有できない」という障害についての推論を行う．今回，5 つのスイッチが図 3 のように接続されており，真ん中のスイッチには VLAN が全く Config されていないが，その他のスイッチには正しく Config されている，という状況を想定した．この状況で，縦に接続されている 4 つのスイッチで VLAN ブロードキャストドメインが共有ができない，と障害情報を入力し，推論を実行する．その結果を図 4 に示す．

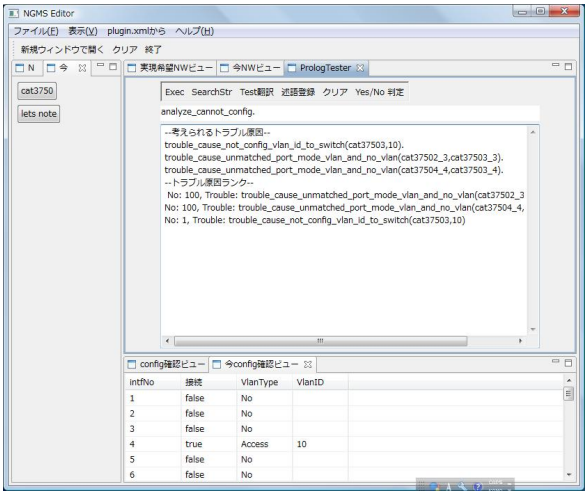

図 4: 推論実行結果

このケースでは，

- スイッチ cat3750-3 に Vlan10 が Config されていない
- スイッチ cat3750-3 とスイッチ cat3750-2 とが接続しているポートの Vlan モードが違う
- スイッチ cat3750-3 とスイッチ cat3750-4 とが接続しているポートの Vlan モードが違う

といったトラブルの原因の可能性が複数抽出されており，その中からスイッチ cat3750-3 自体に Vlan10 が Config されていない，という原因が 1 番に順位付けされている．

最終的に推定された原因がスイッチ cat3750-3 に原因があると判断されると，システムが GUI 上に問題のある箇所の色を変え，ユーザに提示する．その結果を図 5 に示す．現在，Prolog の述語及び，色を変えることによってしか原因推定結果を提示していないが，今後，ユーザに分かりやすい原因推定結果の提示方法を検討する必要がある．

4　関連研究

VLAN の正しい設定を支援するための研究として，櫻田の研究 [3] がある．この研究手法では，モデル検査を用いてネットワーク機器の設定の誤りを検知している．ただし，ネットワーク機器の設定の誤りは検知できるが，物理的な障害は検知出来ない．本研究の手法であれば，ルールに物理的

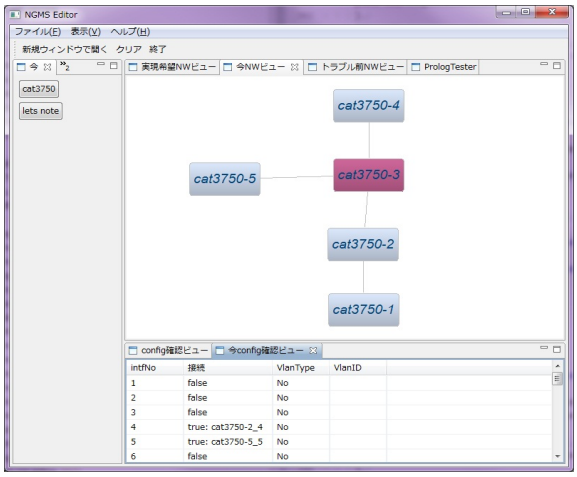

図 5: 原因推定結果の表示

な障害原因を記述することにより，物理的な障害を扱うことも可能である．

また，物理的な障害箇所を特定する研究として，上手らの研究 [5] がある．この研究では，障害箇所を特定するアルゴリズムが提案されており，主に自然災害による物理障害が起きた際の障害部位の特定に焦点を当てている．

園田ら [4] は，テーブルオーバーレイ方式と呼ぶ方法を開発し，これに基づき VLAN のブロードキャストドメインや物理障害が起こった時の障害範囲を把握出来，正常時とどこが違うのかを提示する機能を持つ．これに対し，本研究の手法の特徴は，ルールが手軽に追記可能であるため，今後の課題となるが，ユーザに利用してもらうごとにシステムを賢くしていく方法やルールを追加してもらうことで他の障害原因にも対応可能である点といえる．

岡山ら [1] は，障害情報に応じて，管理者が打ち込む必要があるコマンドをあらかじめシステムにスクリプトとして記述しておき，障害を検知するとそれを実行し，トラブルチケットシステムを組み合わせることによって復旧作業の効率化を図るという手法を提案している．これに対し，本研究の手法の特徴は，直接トラブルの原因を記述することにより，トラブルの原因をユーザにそのまま提示可能である点といえる．

また，VLAN 設定の自動化や統一化を図る技術として，VTP(VLAN Trunking Protocol) や VGRP(GARP VLAN Registration Protocl) といったプロトコルが存在する．これに対し，本研究の手法であれば，物理的な障害についても対応可能である．

本研究の特徴は，ネットワークに関する知識やトラブルに関する知識について，述語を自由に設計出来るため，様々なトラブルの原因に関して原因推定を行える可能性があることといえる．また，現在，実現してはいないが，ルールを手軽に追加可能であるため，ユーザに利用してもらうことでルールが追加され，原因推定精度を上げる仕組みを検討できる．

5　おわりに

本稿では，ネットワークやトラブルに関する知識を知識ベースとしてプログラムに記述しておき，それを基に 2 段階の推論によってトラブルの原因を推定する手法を提案した．トラブルの原因の例として，VLAN の Config 間違いについて焦点を当てた．

今後の課題として，2.1 で述べたトラブルの原因全てに対応出来るようルールを充実させていくことや，本システムを利用しているユーザが，遭遇したトラブルに関するルールを記述出来たり，利用していくと自動的にシステムの原因推定精度が上がっていく等，記述済みのルールを補う仕組みの検討があげられる．また，本稿のケースでは，全ての Config 情報が取得可能であったことを前提としているが，実際にはそうではないケースも考えられるため，実現希望のネットワークと障害情報及び不十分な実ネットワークの情報から原因を推定するためのルールの検討があげられる．

参考文献

1) 岡山聖彦, 山口英, 宮原秀雄:”作業のスクリプト記述に基づいたネットワーク管理支援システム SPLICE/NM の設計と実装”, 電子情報通信学会論文誌,vol.J81-D-1,No.8,pp.1014-1023(1998)

2) Enrico Denti, Andrea Omicini and Alessandro Ricci: ”tuProlog: A Light-Weight Prolog for Internet Applications and Infrastructures”, PRACTICAL ASPECTS OF DECLARATIVE LANGUAGES Lecture Notes in Computer Science, Volume 1990/2001, pp.184-198(2001)

3) 櫻田英樹:”モデル検査を用いたタグ VLAN の設定検査”, 情報処理学会論文誌,vol.47,No.7,pp.2247-2257(2006)

4) 園田健太郎, 松田勝志:”ハッシュ型テーブルオーバレイ方式によるセキュア VLAN

分析システム”, 情報処理学会研究報告,DSM,2008(23),pp.13-18(2008)

5) 上手祐治, 玉川玲, 阿部新:”大規模ネットワークに適した自動障害部位特定アルゴリズムの実装方式と性能評価”, 信学技報.ICM,109(120),pp.7-12(2009)

6) NGMS http://ngms.info/

7) Eclipse Rich Client Platform
http://wiki.eclipse.org/index.php/Rich_Client_Platform

8) Zest: The Eclipse Visualization Toolkit
http://www.eclipse.org/gef/zest/

付録 A VLANConfig 間違い原因抽出ルール

rule1

IF 2 つのスイッチのうち，どちらかに指定の VlanID が登録されていない
THEN その 2 つのスイッチでは VlanID のブロードキャストドメインを共有できない.

rule1 の述語

```
cannot_share_vlan_to_switch_switch(Switch1,_,
  Vlan_id):-
  now_config_vlan_to_switch(Switch1,Vlan_list),
  not(member(Vlan_id,Vlan_list)).
```

rule2

IF 2 つのスイッチが接続されているポートは，トランクポート同士だけれど、どちらかのポートに指定の VlanID が登録されていない
THEN その 2 つのスイッチでは VlanID のブロードキャストドメインを共有できない.

rule2 の述語

```
cannot_share_vlan_port_port(Port1, Port2,
  Vlan_id) :-  now_config_tag(Port1),
  now_config_tag(Port2),
  now_config_tag_vlan_id_to_port(Port1,
  Vlan_id_list), not(member(Vlan_id,
  Vlan_id_list)).
```

rule3

IF Port1 と Port2 がトランクポートとアンタグポートで接続されている
THEN Port1 と Port2 では，Vlan_id のブロードキャストドメインを共有出来ない

rule3 の述語

```
cannot_share_vlan_to_port_port(Port1,
  Port2, Vlan_id) :-
  now_config_untag(Port1),
  now_config_tag(Port2),
  my_trouble_assert(trouble_cause_unmatched_
  vlan_mode_to_port_port(Port1, Port2)).
```

rule4

IF Switch1 と Switch2 の両方に Vlan_id が登録されていない
THEN Switch1 と Switch2 では，Vlan_id のブロードキャストドメインを共有出来ない

rule4 の述語

```
exec_trouble_rank :-
  fail_info_cannot_share_vlan_to_switches(_,_),
  trouble_cause_not_config_vlan_id_to_switch
  (Switch,Vlan_id),my_trouble_rank_assert(1,
  trouble_cause_not_config_vlan_id_to_switch
  (Switch,Vlan_id)).
```

rule5

IF Port1 と Port2 がアクセスポート同士で接続されており，Vlan_id がどちらかに登録されていない
THEN Port1 と Port2 では，Vlan_id のブロードキャストドメインを共有出来ない

rule5 の述語

cannot_share_vlan_port_port(Port1, Port2, Vlan_id) :- now_config_untag(Port1), now_config_untag(Port2),now_config_untag_vlan_id_to_port(Port1,Vlan_id_config), not(Vlan_id=Vlan_id_config), my_trouble_assert(trouble_cause_not_config_vlan_id_to_port(Port1,Vlan_id)).

rule6

IF Port1 と Port2 がトランクポートと Vlan モードになっていない状態で接続されている
THEN Port1 と Port2 では，Vlan_id のブロードキャストドメインを共有出来ない

rule6 の述語

cannot_share_vlan_port_port(Port1,Port2, Vlan_id) :- now_config_tag(Port1), not(now_config_untag(Port2)), not(now_config_tag(Port2)), my_trouble_assert(trouble_cause_unmatched_port_mode_vlan_and_no_vlan(Port1,Port2)).

rule7

IF Port1 と Port2 がトランクポートと Vlan モードになっていない状態で接続されている
THEN Port1 と Port2 では，Vlan_id のブロードキャストドメインを共有出来ない

rule7 の述語

cannot_share_vlan_port_port(Port1,Port2, Vlan_id) :- now_config_untag(Port1), not(now_config_untag(Port2)),not(now_config_tag(Port2)),my_trouble_assert(trouble_cause_unmatched_port_mode_vlan_and_no_vlan (Port1,Port2)).

rule8

IF Port1 と Port2 が両方 Vlan モードになっていない状態で接続されている
THEN Port1 と Port2 では，Vlan_id のブロードキャストドメインを共有出来ない

rule8 の述語

cannot_share_vlan_port_port(Port1,Port2, Vlan_id) :- not(now_config_untag(Port2)),not(now_config_tag(Port2)),not(now_config_untag(Port1)), not(now_config_tag(Port1)),my_trouble_assert (trouble_cause_no_vlan_config_mode_to_both_port_port(Port1,Port2)).