SMT問題の構造と

SMTソルバの効率の相関について

指導教員 : 寺内　多智弘

早稲田大学　基幹理工学部

情報理工学科　寺内研究室

1w153084-8

竹井　友利

2021年2月1日

要旨

充足可能性問題（satisfiability problem, SAT問題）とは、命題論理式が与えられたとき、それに含まれる命題変数の値を真（true）,偽(false)に割り当てることで、命題論理式の値を真にできるか、という問題である。SATソルバとは、SAT問題の充足可能性を判定するものである。

近年の研究[1]で、SATソルバの問題を解く効率（SATソルバの実行時間）とSAT問題からつくるグラフのモジュラリティの間の相関性が指摘されている。

　本研究では、SAT問題で確認されている相関性が、SMT問題に関しても確認できるかどうかを検証した。SMT問題（Satisfiability Modulo Theories）とは、SAT問題の命題変数を述語論理に置き換えたものである。

　実験の結果、SAT問題で指摘されている相関性がSMT問題に関しても確認できた。

目次

[1. はじめに 1](#_Toc62328533)

[2. 準備 2](#_Toc62328534)

[2.1. SAT問題[2][3] 2](#_Toc62328535)

[2.2. SATソルバ[2][3] 2](#_Toc62328536)

[2.3. SMT問題[2][3] 2](#_Toc62328537)

[2.4. SMTソルバ[2][3] 3](#_Toc62328538)

[2.5. Tseitin変換[4] 3](#_Toc62328539)

[2.6. コミュニティ構造 4](#_Toc62328540)

[2.7. モジュラリティ[5] 4](#_Toc62328541)

[3. SMTソルバの効率とモジュラリティの相関性 7](#_Toc62328542)

[3.1. 実験手法 7](#_Toc62328543)

[3.2. 実験環境 9](#_Toc62328544)

[3.3. 実験結果 10](#_Toc62328545)

[4. SMTソルバの効率とモジュラリティの相関性（重みづけあり） 11](#_Toc62328546)

[4.1. 実験手法 11](#_Toc62328547)

[4.2. 実験環境 13](#_Toc62328548)

[4.3. 実験結果 13](#_Toc62328549)

[5. 考察 15](#_Toc62328550)

[6. 今後の課題 16](#_Toc62328551)

[謝辞 17](#_Toc62328552)

[参考文献 18](#_Toc62328553)

[付録 19](#_Toc62328554)

1. はじめに

　現実世界の様々な問題（ソフトウェア・ハードウェア検証、プランニング）をSAT/SMT問題へ置き換えて、ソルバを用いて解くアプローチが普及している。[2]

そのため、SAT/SMT問題の実行速度と問題の構造の相関性について分析するのは有意義でないかと考える。

　本研究では、既存研究で確認されているSAT問題からつくるグラフのモジュラリティとSATソルバの実行時間の相関性が、SMT問題についても、確認できるかを検証する。

1. 準備
   1. SAT問題[2][3]

　充足可能性問題（satisfiablity problem）とは、１つの命題論理式が与えられたとき、その式を満たす、論理式全体が真になるようなそれぞれの命題変数への真偽の割り当てが存在するかということをいう。

例.

A = false, B=true, C = trueを代入すると、論理式全体は真になる。よってこの式は充足可能である。

* 1. SATソルバ[2][3]

　SAT問題を受け取り、SMT問題の充足可能性を判定するためのプログラムである。SATソルバは、連言標準形（Conjunctive Normal Form, CNF）のSAT問題のみを入力として受け取る。

* 1. SMT問題[2][3]

　SMT問題（Satisfiability Modulo Theories）とは、SAT問題が命題論理を扱うのに対して、述語論理を扱う問題のこと。したがって、SMT問題は関数や算術演算子を扱うことができるため、SAT問題より豊かな表現力を持つ。

例.

a = 3, b = 6を代入すると、論理式全体は真になる。よってこの式は充足可能である。

* 1. SMTソルバ[2][3]

　SMTソルバはSMT問題の充足可能性を判定するためのプログラムである。SMTソルバのアプローチとしてEager,Lazyの2種のアプローチがある。

　Eagerアプローチは、SMT問題をすべて命題論理の充足問題に変換して全体をSATソルバで解く手法である。このアプローチは、各理論の論理式を、充足可能性が等価である命題論理の論理式に変換する必要がある。この変換には、論理毎の技法が必要で、必ず充足可能性を維持した変換ができるとは限らない。また、変換された命題論理式が長くなる可能性がある。

　Lazyアプローチは、SATソルバと理論ソルバを利用してSMT問題を解くという手法である。一般的なSMTソルバはLazyアプローチである。本実験で用いたSMTソルバもこのLazyアプローチである。

* 1. Tseitin変換[4]

　Tseitin変換は1968年にG. S. Tseitinによって導入された充足同値な変換方法である。この変換方法により、比例したサイズのCNFを求めることが可能である。LazyアプローチのSMTソルバはCNFの論理式のみを入力として受け取るSATソルバを用いるので、SMT問題がCNFでないときにこの変換を行っている。

まず、 をAで表す。

を新たな命題変数rで置き換え、同時にｒとが同値であることを表す論理式を追加する。同様に、についても新しい命題変数ｓに置き換えると、以下の論理式A’が得られる。

さらに、変形すると、

となる。

例.

n=100, m=100のとき

となり、選言節は合計:1 + (1+ n) + (1 + m) = 203個となる。

一方、分配法則の場合、選言節の合計はnm個になるので10000個となる。

* 1. コミュニティ構造

　コミュニティ構造とは、グラフ中のノードがいくつかのまとまりに分割したものを示す。

高いコミュニティ構造を持つとは、各コミュニティ内では密なつながりを持ち、コミュニティ外とのつながりが疎である構造を持つということ。

* 1. モジュラリティ[5]

モジュラリティは、ネットワークの与えられた分割に対して、「同じグループに属するノード同士が繋がるリンク数の全リンク数に占める割合」から「リンクがランダムに配置された場合のグループ内のリンク数の全体のリンク数に対する割合の期待値」を引いた値として定義される。

ノードの数N、リンクの数Mのネットワークを考える。ノードをのようにC個のグループに分ける。ネットワークの隣接行列をAとして、行列成分はノードr,s間に存在するリンク数とする。なお、本実験では、無向グラフかつリンクへの重みづけを考えないので、Aの各成分は１か０のみになり、Aは対称行列となる。したがって、隣接行列の成分の和は

となる。

また、に属するノードと、に属するノードがつながるリンク数の合計の全リンク数に占める割合は以下のように書ける。

したがって、「同じグループに属するノード同士が繋がるリンク数の全リンク数に占める割合」はで表せる。

次に、「リンクがランダムに配置された場合のグループ内のリンク数の全体がリンク数に対する割合の期待値」について考える。

まず、M本のリンクを半分に切り、2M本の「手」（リンクの半分）を作る。次に、ノードに接続した「手」をランダムに２つずつ選択してつなぎなおす。ここで、ノードrの「手」の数をノードrの次数と呼び、以下のように表せる。

全ノードの次数の合計は

となる。上記のようにランダムにリンクをつなぐと、リンクの一方に内のノードの「手」が選ばれる確率は以下のように書ける。

これは、少なくとも一方が内のノードにつながれるリンク数の全リンク数に占める割合である。リンクの両端が内のノードである場合の期待値は、となる。

従って、モジュラリティは以下のように定義できる。

例.

図2.1のようなネットワークがあったとするとモジュラリティは以下のように求める。

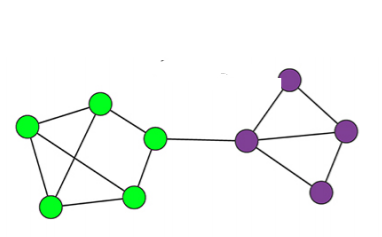


図2.1 ネットワークの例

まず、緑のグループを、紫ｎグループをとする。

するとモジュラリティQは

となる。それぞれ求めていくと

となり、このようにモジュラリティは求められる。

1. SMTソルバの効率とモジュラリティの相関性
   1. 実験手法

　SMTソルバがSMT問題を解く時間を計測する。SMT問題からグラフを作成して、そのグラフのモジュラリティを求める。これを複数の問題に対して計測して、相関性があるか確認する。SMTソルバには、Z3[6]を用いる。モジュラリティ計算にはGephi[7]というツールを使用する。

ベンチマークは<http://smtlib.cs.uiowa.edu/benchmarks.shtml>[8]にあるQF\_LIAを利用する。

具体的な実験手順は以下に従う。

1. SMT問題から以下のようにグラフを構築する。図3.1参照。

・頂点 : 各変数を頂点とする

・辺 : SMT問題の論理式からCNFに変換した後の節内の変数を辺で結ぶ。

1. グラフにおけるモジュラリティをGephiで計算する。

図3.1のグラフをcsv形式のノードファイル(図3.2参照)とcsv形式のエッジファイル(図3.3参照)で表現したものをGephiは入力として受け取る。

1. Z3を用いて、実行時間を計測する。
2. 図3.4より、実行時間とベンチマークのデータ量が比例関係にあることが推測できるので、各ベンチマークのデータ量1000kbであると仮定して実行時間を変換する。
3. 変換した実行時間とモジュラリティの相関性を調べる。表3.1、図3.5

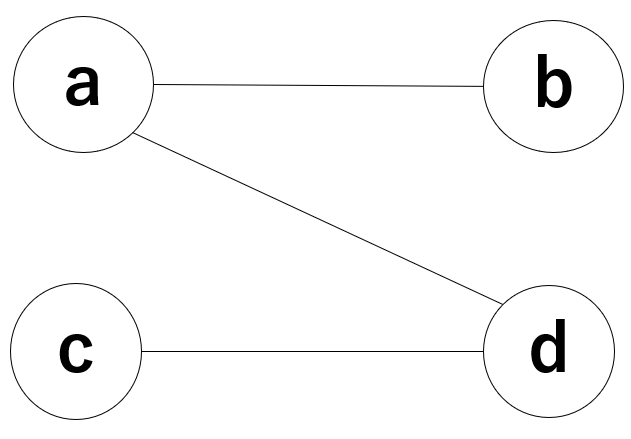


図3.1 グラフ構築の例

|  |
| --- |
| Id,Label  0,a  1,b  2,c  3,d |

図3.2 グラフ（図3.1）を表すノードファイル

|  |
| --- |
| Source,Target  0,1  0,2  2,3 |

図3.3 グラフ(図3.1)を表すエッジファイル

図3.4 実行時間とベンチマークのデータ量の関係

* 1. 実験環境

デバイス名:DESKTOP-N7AGLEM

CPU: intel core i5-8299U CPU @2.50GHz 2.70GHz

実装RAM: 8.00 GB

* 1. 実験結果

　モジュラリティの値とZ３の実行時間をそれぞれ表3.1に示す。

また、それをグラフに図示したものを図3.5に示す。

表3.1　SMTソルバの実行時間(変換後)とモジュラリティの関係

|  |  |  |
| --- | --- | --- |
| ソースファイル名 | 実行時間（変換後） [s] | モジュラリティ |
| ParallelPrefixSum\_live\_bgmc000 | 0.685621971 | 0.53 |
| ParallelPrefixSum\_live\_bgmc002 | 0.519145161 | 0.619 |
| ParallelPrefixSum\_live\_bgmc003 | 0.442586387 | 0.659 |
| ParallelPrefixSum\_live\_blmc000 | 0.515041451 | 0.679 |
| ParallelPrefixSum\_live\_blmc002 | 0.448232323 | 0.806 |
| ParallelPrefixSum\_safe\_bgmc002 | 0.630972818 | 0.583 |
| ParallelPrefixSum\_safe\_bgmc003 | 0.528691176 | 0.651 |
| ParallelPrefixSum\_safe\_bgmc004 | 0.456772947 | 0.681 |
| ParallelPrefixSum\_safe\_bgmc005 | 0.462439286 | 0.76 |
| ParallelPrefixSum\_safe\_bgmc006 | 0.457845714 | 0.611 |
| ParallelPrefixSum\_safe\_blmc000 | 0.519452555 | 0.624 |
| ParallelPrefixSum\_safe\_blmc001 | 0.462571429 | 0.725 |
| ParallelPrefixSum\_safe\_blmc002 | 0.485787179 | 0.753 |
| ParallelPrefixSum\_safe\_blmc003 | 0.474315068 | 0.798 |
| ParallelPrefixSum\_safe\_blmc004 | 0.470299861 | 0.813 |
| ParallelPrefixSum\_safe\_blmc005 | 0.46613789 | 0.819 |
| ParallelPrefixSum\_safe\_blmc006 | 0.51002963 | 0.837 |

図3.5　SMTソルバの実行時間とモジュラリティの関係

1. SMTソルバの効率とモジュラリティの相関性（重みづけあり）
   1. 実験手法

　SMTソルバがSMT問題を解く時間を計測する。SMT問題からグラフを作成して、そのグラフのモジュラリティを求める。これを複数の問題に対して計測して、相関性があるか確認する。SMTソルバには、Z3[6]を用いる。モジュラリティ計算にはGephi[7]というツールを使用する。

ベンチマークは<http://smtlib.cs.uiowa.edu/benchmarks.shtml>[8]にあるQF\_LIAを利用する。重みづけの効果を検証したいので、第３章と同じベンチマークを使用する。

具体的な実験手順は以下に従う。

1. SMT問題から以下のようにグラフを構築する。図3.1参照。

・頂点 : 各変数を頂点とする

・辺 : SMT問題の論理式からCNFに変換した後の節内の変数を辺で結ぶ。

辺があらわれるたびに、辺の重みを１加算していく。

1. グラフにおけるモジュラリティをGephiで計算する。

図4.1のグラフをcsv形式のノードファイル(図4.2参照)とcsv形式のエッジファイル(図4.3参照)で表現したものをGephiは入力として受け取る。

1. Z3を用いて、実行時間を計測する。
2. 図3.4より、実行時間とベンチマークのデータ量が比例関係にあることが推測できるので、各ベンチマークのデータ量1000kbであると仮定して実行時間を変換する。
3. 変換した実行時間とモジュラリティの相関性を調べる。表4.1、図4.5

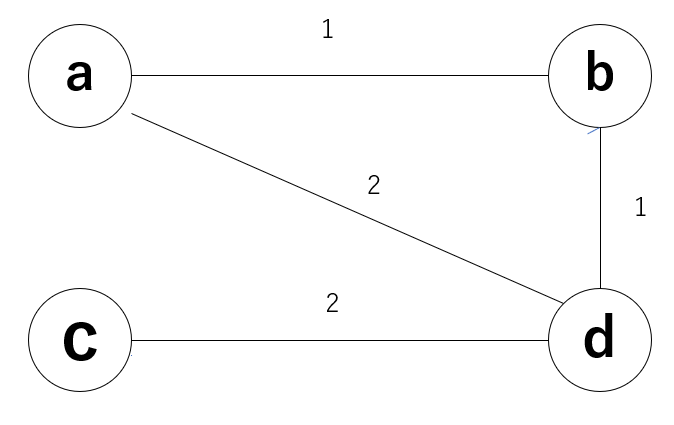


図4.1 グラフ構築の例

の場合

|  |
| --- |
| Id,Label  0,a  1,b  2,c  3,d |

図4.2 グラフ（図4.1）を表すノードファイル

|  |
| --- |
| Source,Target,Weight  0,1,1  0,3,2  1,3,1  2,3,2 |

図4.3 グラフ(図4.1)を表すエッジファイル

* 1. 実験環境

デバイス名:DESKTOP-N7AGLEM

CPU: intel core i5-8299U CPU @2.50GHz 2.70GHz

実装RAM: 8.00 GB

* 1. 実験結果

　モジュラリティの値とZ３の実行時間をそれぞれ表4.1に示す。

また、それをグラフに図示したものを図4.4に示す。

表4.1　SMTソルバの実行時間(変換後)とモジュラリティの関係

|  |  |  |
| --- | --- | --- |
| ソースファイル名 | 実行時間（変換後） [s] | モジュラリティ |
| ParallelPrefixSum\_live\_bgmc000 | 0.685621971 | 0.617 |
| ParallelPrefixSum\_live\_bgmc002 | 0.519145161 | 0.73 |
| ParallelPrefixSum\_live\_bgmc003 | 0.442586387 | 0.768 |
| ParallelPrefixSum\_live\_blmc000 | 0.515041451 | 0.787 |
| ParallelPrefixSum\_live\_blmc002 | 0.448232323 | 0.826 |
| ParallelPrefixSum\_safe\_bgmc002 | 0.630972818 | 0.591 |
| ParallelPrefixSum\_safe\_bgmc003 | 0.528691176 | 0.709 |
| ParallelPrefixSum\_safe\_bgmc004 | 0.456772947 | 0.75 |
| ParallelPrefixSum\_safe\_bgmc005 | 0.462439286 | 0.765 |
| ParallelPrefixSum\_safe\_bgmc006 | 0.457845714 | 0.787 |
| ParallelPrefixSum\_safe\_blmc000 | 0.519452555 | 0.69 |
| ParallelPrefixSum\_safe\_blmc001 | 0.462571429 | 0.752 |
| ParallelPrefixSum\_safe\_blmc002 | 0.485787179 | 0.751 |
| ParallelPrefixSum\_safe\_blmc003 | 0.474315068 | 0.772 |
| ParallelPrefixSum\_safe\_blmc004 | 0.470299861 | 0.796 |
| ParallelPrefixSum\_safe\_blmc005 | 0.46613789 | 0.825 |
| ParallelPrefixSum\_safe\_blmc006 | 0.51002963 | 0.83 |

図4.4　SMTソルバの実行時間とモジュラリティの関係

1. 考察

図3.5,図4,4を参照すると、SMTソルバの実行時間とモジュラリティの相関性を確認できた。モジュラリティが大きくなると、実行時間は短くなり、小さくなると実行時間は長くなるという相関性であることがわかる。

また、重みづけ有りの方が、近似直線と測定値の差が小さくなり、より相関性があると考えられる。

1. 今後の課題

本実験では、適切なベンチマークを17個しか見つけられなかった。

以下の原因が考えられる。

１つ目の原因は、実験環境に十分な演算能力がなかったことである。２つ目の原因は、適当なベンチマークを見つけられたが作成したグラフを作成するプログラムがベンチマーク内のite演算を処理できなかったことである。

今後の課題として、高い演算能力を持つCPUでの実験を行うこと、グラフ作成プログラムの改善、プログラムの改善が考えられる。

また、重みづけの方法として、節の中で複数の辺を構成できるとき（の場合、c,d間の辺の重みは２）も、カウントしたが、節の中はカウントしない、重みを変えるなどいろいろな重みづけをして実験をして、最も実行時間と相関性のある重みづけを探りたい。

謝辞

　本研究を進めるにあたり、ご指導いただきました早稲田大学情報理工学科の寺内 多智弘教授に心より深く感謝致します。また、多くの助言をいただきました寺内研究室の山内君に心より深く感謝いたします。

参考文献

[1] C. Ans´otegui, J. Gir´aldez-Cru, and J. Levy,

“The community structure of SAT formulas”, SAT 2012, pp.410-423, 2012.

[2]「SAT/SMTソルバのしくみ」< <https://www.slideshare.net/sakai/satsmt> >

2021/01/02 アクセス.

[3]「SATソルバ・SMTソルバの技術と応用」

< <https://www.jstage.jst.go.jp/article/jssst/27/3/27_3_3_24/_pdf> > 2021/01/02 アクセス.

[4]「ソフトウェア科学特論：命題論理」< <https://tamura70.gitlab.io/lect-proplogic/org/proplogic.html> >2021/01/02アクセス.

[5] Newman, M. E. J. (2006). “Fast algorithm for detecting community structure in networks”.

[6]「Z3」, < https://github.com/Z3Prover/z3 >, 2021/01/02 アクセス.

[7]「Gephi – The Open Graph Viz Platform」, < https://gephi.org/ >, 2021/01/02 アクセス.

[8] 「SMT-LIB Benchmarks」, < http://smtlib.cs.uiowa.edu/benchmarks.shtml >, 2021/01/02 アクセス.

付録

SMTソルバの実行時間を求めるプログラム

|  |
| --- |
| open Z3  let smtlib2\_to\_assertions (ctx: Z3.context) (smt2file\_name : string) : Z3.Expr.expr list =  let astvec = SMT.parse\_smtlib2\_file ctx smt2file\_name [] [] [] [] in  Z3.AST.ASTVector.to\_expr\_list astvec  let to\_string\_exprs (expr\_list : Z3.Expr.expr list) : string =  List.fold\_left(fun acm e -> Printf.sprintf "%s%s:\n" acm (Z3.Expr.to\_string e)) "" expr\_list  let print\_exprs (expr\_list : Z3.Expr.expr list) : unit =  Printf.printf "%s\n" (to\_string\_exprs expr\_list)  let is\_satisfiable (ctx :Z3.context) (exprs : Z3.Expr.expr list) : Z3.Solver.status =  let solver = Z3.Solver.mk\_solver ctx None in  Z3.Solver.check solver exprs  let get\_time f ctx (exprs : Expr.expr list) : Z3.Solver.status =  let t = Sys.time() in  let f\_ctx\_exprs = f ctx exprs in  Printf.printf "Execution time : %fs\n" (Sys.time() -. t);  f\_ctx\_exprs  let \_ =  let ctx = Z3.mk\_context [("model","true");("proof","true")] in  let argument1 =  try List.hd (List.tl (Array.to\_list Sys.argv)) with  Failure \_ -> (Printf.printf "Usage:./a.out file.smt2\n"); exit 1  in  let exprs = smtlib2\_to\_assertions ctx argument1 in  Printf.printf "%s\n" (Z3.Solver.string\_of\_status (get\_time is\_satisfiable ctx exprs)); |

SMT問題からcsv形式のノードファイルとエッジファイルを出力するプログラム

|  |
| --- |
| open Z3  type node = {id : int; label : string}  let to\_string\_exprs (expr\_list : Z3.Expr.expr list) : string =  List.fold\_left (fun acm e -> Printf.sprintf "%s%s," acm (Z3.Expr.to\_string e)) "" expr\_list ^ "\n"  let print\_exprs (expr\_list : Z3.Expr.expr list) : unit =  Printf.printf "%s" (to\_string\_exprs expr\_list)  let rec get\_all\_variable (result\_list : Expr.expr list) (elem\_list : Expr.expr list) : Expr.expr list =  match elem\_list with  [] -> result\_list  | e :: elem\_list ->  let getArgs\_e : Expr.expr list = Expr.get\_args (e : Expr.expr) in  if (List.length (getArgs\_e : Expr.expr list) > 0) then begin  let res : Expr.expr list = get\_all\_variable result\_list getArgs\_e in  get\_all\_variable res elem\_list  end  else  if(Expr.is\_numeral (e : Expr.expr)) then  get\_all\_variable result\_list elem\_list  else begin  (\*append e to res\_list \*)  (\* [e] :: result\_list \*)  let val\_list = List.append [e] result\_list in  get\_all\_variable val\_list elem\_list  end  let rec print\_all\_variable (elem\_list : Expr.expr list) : unit =  match elem\_list with  [] -> ()  | e :: elem\_list ->  let getArgs\_e :Expr.expr list = Expr.get\_args (e :Expr.expr) in  if ( Expr.is\_numeral (e : Expr.expr) ) then  print\_all\_variable elem\_list  else  (\* let getArgs\_e :Expr.expr list = Expr.get\_args (e :Expr.expr) in \*)  if (List.length (getArgs\_e : Expr.expr list) > 0) then begin  print\_all\_variable getArgs\_e;  print\_all\_variable elem\_list  end  else  Printf.printf "variable : %s\n" ( Expr.to\_string (e : Expr.expr) )  let rec print\_astvec (res\_channel : out\_channel) (expr\_list : Expr.expr list) : unit =  match expr\_list with  [] -> ()  | expr :: expr\_list ->  let args\_of\_expr = Expr.get\_args expr in  let res\_expr\_list : Expr.expr list = get\_all\_variable [] args\_of\_expr in  print\_exprs (res\_expr\_list);  output\_string res\_channel (to\_string\_exprs (res\_expr\_list));  print\_astvec res\_channel expr\_list  let rec is\_already\_exist (node\_list : node list) (e : string ) : bool =  match node\_list with  [] -> false  | n :: node\_list ->  if (String.compare n.label e = 0) then  true  else  is\_already\_exist node\_list e  let rec set\_node (res\_list : node list) (i : int) (expr\_list : Expr.expr list) : node list =  match expr\_list with  [] -> res\_list  | expr :: expr\_list ->  if (is\_already\_exist res\_list (Expr.to\_string expr)) then  set\_node res\_list i expr\_list  else  let res\_node\_string = Expr.to\_string expr in  let n : node = {id = i; label = res\_node\_string} in  let after\_res\_list : node list = List.append [n] res\_list in  set\_node after\_res\_list (i+1) expr\_list  let rec get\_node\_list (res : node list) (i : int) (expr\_list : Expr.expr list) : node list =  match expr\_list with  [] -> res  | expr :: expr\_list ->  let args\_of\_expr = Expr.get\_args expr in  let res\_expr\_list : Expr.expr list = get\_all\_variable [] args\_of\_expr in  if (List.length res\_expr\_list > 1) then begin (\*set node info from variables\*)  let res\_node\_list : node list = set\_node res i res\_expr\_list in  get\_node\_list res\_node\_list (List.length res\_node\_list) expr\_list  end  else  get\_node\_list res i expr\_list  let rec get\_id (node\_list : node list) (l : string) : int =  match node\_list with  [] -> (-1)  | n :: node\_list ->  if(String.compare n.label l = 0) then  n.id  else  get\_id node\_list l  let rec get\_var\_of\_node\_id (res : int list) (node\_list : node list) (var\_list : Expr.expr list) : int list =  match var\_list with  [] -> res  | var :: var\_list ->  let res\_ = List.append [(get\_id node\_list (Expr.to\_string var))] res in  get\_var\_of\_node\_id res\_ node\_list var\_list  let rec put\_edge (matrix : int array array) (i : int) (node\_id : int list) : int array array =  match node\_id with  [] -> matrix  | j :: node\_id ->  matrix.(i).(j) <- 1;  put\_edge matrix i node\_id  let rec set\_edge (matrix : int array array) (node\_id : int list) : int array array =  match node\_id with  [] -> matrix  | i :: node\_id ->  let matrix\_ = put\_edge matrix i node\_id in  set\_edge matrix\_ node\_id  let rec set\_matrix (matrix : int array array) (node\_list : node list) (expr\_list : Expr.expr list) :int array array =  match expr\_list with  [] -> matrix  | expr :: expr\_list ->  let args\_of\_expr = Expr.get\_args expr in  let res\_expr\_list : Expr.expr list = get\_all\_variable [] args\_of\_expr in  if (List.length res\_expr\_list > 1) then (\*if there is edge\*)  let node\_id : int list = get\_var\_of\_node\_id [] node\_list res\_expr\_list in  let added\_matrix : int array array = set\_edge matrix node\_id in  set\_matrix added\_matrix node\_list expr\_list  else  set\_matrix matrix node\_list expr\_list  let rec print\_edge (res\_edge\_channel : out\_channel) (i : int) (j : int) (num\_of\_node : int) (matrix : int array array) : unit =  if(j = num\_of\_node)then  ()  else begin  if(matrix.(i).(j) = 1) then begin  output\_string res\_edge\_channel (Printf.sprintf "%d,%d\n" i j);  print\_edge res\_edge\_channel i (j+1) num\_of\_node matrix  end  else  print\_edge res\_edge\_channel i (j+1) num\_of\_node matrix  end  let rec call\_print\_edge (res\_edge\_channel : out\_channel) (gyou\_i : int) (num\_of\_node : int) (matrix : int array array) : unit =  if(gyou\_i = num\_of\_node) then  ()  else begin  print\_edge res\_edge\_channel gyou\_i 0 num\_of\_node matrix;  call\_print\_edge res\_edge\_channel (gyou\_i + 1) num\_of\_node matrix  end  let rec print\_node (res\_node\_channel : out\_channel) (node\_list : node list) : unit =  match node\_list with  [] -> ()  | n :: node\_list ->  output\_string res\_node\_channel (Printf.sprintf "%s,%s\n" (string\_of\_int n.id) n.label);  print\_node res\_node\_channel node\_list  let convert\_to\_cnf (ctx:Z3.context) (exprs : Z3.Expr.expr list) : Z3.Expr.expr list =  let goal = Z3.Goal.mk\_goal ctx false false false in  Z3.Goal.add goal exprs;  let tseitin\_tactic = Z3.Tactic.mk\_tactic ctx "tseitin-cnf" in  let apply\_result : Z3.Tactic.ApplyResult.apply\_result = Z3.Tactic.apply tseitin\_tactic goal None in  Printf.printf "num\_of\_subgoals = %d\n" (Z3.Tactic.ApplyResult.get\_num\_subgoals apply\_result);  let goal\_ : Z3.Goal.goal = Z3.Tactic.ApplyResult.get\_subgoal apply\_result 0 in  Z3.Goal.get\_formulas goal\_  let \_ =  let ctx = Z3.mk\_context [("model", "true");("proof","true")] in  let argument1 =  try List.hd (List.tl (Array.to\_list Sys.argv)) with  Failure \_ -> (Printf.printf "Usage:./a.out file.smt2\n"); exit 1  in  let astvec = SMT.parse\_smtlib2\_file ctx argument1 [] [] [] [] in  (\*  let astVecString = AST.ASTVector.to\_string astvec in  Printf.printf "astVecString %s\n" astVecString;  \*)  let exprList\_of\_astvec : Expr.expr list = AST.ASTVector.to\_expr\_list astvec in  let exprList\_after\_tseitin : Expr.expr list = convert\_to\_cnf ctx exprList\_of\_astvec in  (\*  let res\_channel : out\_channel = open\_out "res\_variable.csv" in  print\_astvec res\_channel exprList\_of\_astvec;  \*)  let node\_list : node list = get\_node\_list [] 0 exprList\_after\_tseitin in  let num\_of\_node : int = List.length node\_list in  Printf.printf "num\_of\_node = %d\n" num\_of\_node;  let inited\_matrix :int array array = Array.make\_matrix num\_of\_node num\_of\_node 0 in  let res\_matrix = set\_matrix inited\_matrix node\_list exprList\_after\_tseitin in  let res\_edge\_channel : out\_channel = open\_out "res\_edge.csv" in  output\_string res\_edge\_channel "Source,Target\n";  call\_print\_edge res\_edge\_channel 0 num\_of\_node res\_matrix;  let res\_node\_channel : out\_channel = open\_out "res\_node.csv" in  output\_string res\_node\_channel "Id,Label\n";  print\_node res\_node\_channel node\_list |