数理解析研究所講究録
1125巻 2000年 129-141

グラフ探索アルゴリズムの形式的検証とモデル検査への応用について

On Formal Verification of Graph Search Algorithms and Its Application to Model Checking

山本 光晴[†] Mitsuharu YAMAMOTO
高橋 孝一[††] Koichi TAKAHASHI
萩谷 昌己[‡] Masami HAGIYA
西崎 真也[‡†] Shin-ya Nishizaki
玉井 哲雄[‡‡] Tetsuo Tamai

†千葉大学 理学部
Faculty of Science, Chiba University
††電子技術総合研究所 情報アーキテクチャ部
Computer Science Division, Electrotechnical Laboratory
‡東京大学大学院 理学系研究科
Graduate School of Science, The University of Tokyo
‡†東京工業大学大学院 情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology
‡‡東京大学大学院 総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo

概要

グラフの探索問題は計算機科学の諸分野で基礎理論として用いられ、対象領域に沿った様々な最適化アルゴリズムが提案されている。これらの最適化アルゴリズムは高度な工夫や複雑な前提条件のためにその正当性が必ずしも自明ではなく、形式的検証の対象として相応しいと言える。証明検証系による形式的検証の効果を最大限に発揮するには、問題領域の適切な抽象化が不可欠である。本講演では、グラフ探索アルゴリズムのあるクラスの抽象化と実際の検証、最適化アルゴリズムの検証への発展について述べる。さらに、状態空間がなすグラフの網羅的探索によって検証を行うモデル検査について、そこで用いられるアルゴリズムの検証への応用に関して述べる。

1 はじめに

間違いのないシステムの設計・実装のために、計算機を用いた形式的手法と呼ばれる技術が様々な分野で利用されている。システムが正しく動くことを検証するために用いられる手法として、モデル検査系と証明検証系とがある。モデル検査は有限の状態空間に対する網羅的探索によって、システムがある性質を満たすことを自動的に検査する手法である。一方、証明検証系では示すべき性質を定理として公理と推論規則から証明することによって、システムの正当性を保証する。

証明検証系とモデル検査にはそれぞれの長所・短所がある。証明検証系は主に矛盾を導き出さないことに重点が置かれており、正当性を保証する技術

が確立している。一方、モデル検査は絶対的な安全性よりも、設計上の誤りを早い段階で検出することに重点が置かれている。解ける問題のサイズに関しては、通常はモデル検査では対象が有限のシステムに限られ、また、有限であったとしてもあまりに大きなサイズのものは解けない。一方、証明検証系では適切に定式化すれば無限のケースでも扱える。最後に、証明検証系では検証にかなりの人間の手の介入が必要であるが、モデル検査は高度に自動化されており、「push button」技術と称されることもある。

モデル検査と証明検証系の双方の欠点を補うため、両者を相補的に用いるための研究がいろいろ行われている。一つは証明検証系の中にモデル検査を決定手続き的に取り込む方法である [13] [2] [1]。この方法は主に解けるサイズの問題と、自動化の問題を対象としている。もう一つはモデル検査で用いられる方法の正当性を証明検証系で検証するというものである。この方法は検証結果の正当性の問題を対象としている。もしモデル検査でのアルゴリズムの正当性が証明検証系で検証されれば、モデル検査の利便性を生かしつつ、モデル検査の検証結果に対してより確信を持つことができる。モデル検査で用いられる手法の形式的検証の関連研究としては、モデル検査系 SPIN [8] [9]で用いられている partial-order reduction という最適化法に関する形式的検証を行った Chou と Peled の仕事 [4]が挙げられる。

モデル検査で用いられる検証アルゴリズムのあるクラスは、グラフ探索アルゴリズムのアルゴリズムと見ることができる。例えば、システムが悪い状態に陥いらないことを保証するためには、初期状態から始めて悪い状態に辿りつく経路が存在しないことを示せばよい。

グラフの探索問題は計算機科学の諸分野で基礎理論として用いられており、その重要性は言うまでもないであろう。もしアルゴリズムの目的がある条件を満たす解を求めることであれば、求まった解が与えられた条件を満たすかどうかを検査すればよい。しかし、与えられた条件を満たす解が存在しないことを保証するのがアルゴリズムの目的であれば、上の方針は適用できず、アルゴリズム自体の正当性を何らかの方法で検証しなければならない。上記のモデル検査の例は正に「解がないことを保証」する例となっている。

また、グラフの探索問題は効率化のため、また、組み合わせ爆発を防ぐために、対象領域に沿った様々な最適化アルゴリズムが提案されている。しかしそれらの最適化手法は高度に工夫されたものが多く、また、凝ったアルゴリズムほど複雑な前提条件を要することもあってそれらの正当性は必ずしも自明でない。このことからもグラフの探索問題が形式的検証の対象として相応しいと言える。そこで、本論文ではグラフの探索問題を対象とし、その証明検証系による形式化手法とモデル検査で用いられるアルゴリズムの検証への応用について論じる。

証明検証系による検証は大変労力を要するため、各々の問題を一つ一つ形式化するのは非常に効率が悪い。そこで、問題やアルゴリズムの**適切な抽象化**を考え、具体的な問題やアルゴリズムに関する正当性は抽象的なアルゴリズムの検証から導き出すようにすることが望ましい。本論文では、探索の対象を束として考える抽象化と、グラフのノードの間に順序関係を導入する抽象化を取り上げ、形式化の手法とどのような問題に適用可能であるかについて述べる。

本論文の構成は以下のようになっている。2節から5節まではグラフの頂点に束の要素を割り当てることにより、グラフの探索問題の抽象化を行い、その形式的検証の結果と考察を述べる。2節ではいくつかの準備の後、問題の設定を行う。3節で抽象グラフ探索アルゴリズムを実際に証明検証系の上で定義し、その上で証明した性質について述べる。4節では抽象グラフ探索アルゴリズムを具体化して得られる A* アルゴリズムについて、その定義と、抽象アルゴリズムの検証結果をどのように利用できるかについて述べる。5節では形式的検証を通して我々が得た知見について解説する。6節ではよりモデル検査アルゴリズムに適した抽象化として、グラフの点に順序関係を導入する方法について述べる。7節では6節で定義したアルゴリズムの具体化の例となっているものを挙げ、抽象アルゴリズム上で示した性質がどのように利用できるかについて述べる。8節では7節までに現れた各種アルゴリズムの間の関係をまとめる。

2 準　備

2.1 グラフアルゴリズム

グラフと束の要素を関連づけて探索を行う方法は、Kildall [10]以来、とくにデータフロー解析の分野で様々な定式化が行われている。本論文では、玉井 [14]の定式化をもとにしており、本節ではこの定式化について簡単に説明する。

まず有向グラフと半束の説明を含めて探索問題の設定について述べ、その後に2種類の解 (不動点解と経路解) について説明する。さらに、この定式化の例となる問題をいくつか挙げる。最後に不動点解を求めるためのアルゴリズムについて述べる。

2.1.1 問題の設定

有向グラフ G は頂点の集合 V と辺の集合 E の組 (V, E) で表される。各辺 $e \in E$ について、辺の始点・終点をそれぞれ、$h(e)$, $t(e)$ で表す。頂点 v に入ってくる辺の集合 $\mathrm{in}(v)$ と、v から出ていく辺の集合 $\mathrm{out}(v)$ は $\mathrm{in}(v) = \{e \in E \mid t(e) = v\}$, $\mathrm{out}(v) = \{e \in E \mid h(e) = v\}$ によって定義される。グラフには検索の開始点となる点 $s \in V$ が決められていて、$\mathrm{in}(s) = \emptyset$ であることを仮定する。もし開始点が複数あったり、開始点に入ってくる辺が存在する場合でも、仮想的な開始点を新しく設け、もとの開始点に仮想的な辺を加えればよい。

グラフ G 中の経路 $p = (v, [e_1, \ldots, e_n])$ $(n \geq 0)$ は隣りあう辺の終点と始点を共有するような辺の列である (辺の列は空であってもよい)。つまり、$v = h(e_1)$ かつ $i = 1, \ldots, n-1$ について $t(e_i) = h(e_{i+1})$ となるものである。頂点 v から頂点 w への経路が存在するとき、w は v から**到達可能**であるという。

L を半束とする。つまり、ベースとなる集合 L と、L の要素の間に交わり $(\wedge)$ の操作が定義されており、以下の性質を満たす。

1. $\forall x, y \in L.\ x \wedge y = y \wedge x$;
2. $\forall x, y, z \in L.\ x \wedge (y \wedge z) = (x \wedge y) \wedge z$;
3. $\forall x.\ x \wedge x = x$.

L 上の半順序関係 $\leq$ は次のように定義される。

$$\forall x, y \in L.\ x \leq y \Longleftrightarrow x \wedge y = x.$$

さらに、L には最大元 $\top$ が存在することを仮定しておく。すなわち、$\forall x \in L.\ x \wedge \top = x$ が成り立つ。

解くべき問題が与えられるとき、各辺 $e \in E$ には L から L への単調関数 f_e が関連づけられる $(\forall x, y \in L.\ x \leq y \Longrightarrow f_e(x) \leq f_e(y))$。ただし、$f_e(x) = \top$ となるのは $x = \top$ のときだけであると仮定する。この関数は自然に経路 $p = (v, [e_1, \ldots, e_n])$ に拡張でき、$f_p = f_{e_n} \circ \cdots \circ f_{e_1}$ となる。

解くべき問題は次のように定式化される。

> グラフ $G = (V, E)$、半束 L、開始点 $s \in V$、開始点 s での値 $b_s \in L \setminus \{\top\}$、各辺 $e \in E$ に対する単調関数 $f_e \in L^L$ の割り当てが与えられたとき、各点 $v \in V$ に対し、ある条件を満たす $x_v \in L$ を求める。

上記の「条件」にどのようなものを指定するかによって2種類の解が定式化されている。以下にそれらの解と種類と対応する条件を述べる。

2.1.2 不動点解

$v \in V$ への半束の要素の割り当て $x_v \in L$ が以下の条件を満たすとき、$x_v \in L$ を**不動点解**と呼ぶ。

$$\begin{cases} x_s = b_s \\ x_v = \bigwedge_{t(e)=v \text{ かつ } h(e) \text{は} s \text{から到達可能}} f_e(x_{h(e)}) \quad (v \neq s) \end{cases}$$

2番目の等式はいくらか不正確で曖昧である。ここでは、「右辺が存在して、かつ x_v に等しい。」ということを表しているとする。(束の完全性やグラフの有限性は仮定していないので、2番目の等式の右辺が必ずしも存在するとは限らない)

2.1.3 経路解

$v \in V$ への半束の要素の割り当て $x_v \in L$ が以下の条件を満たすとき、$x_v \in L$ を**経路解**と呼ぶ。

$$x_v = \bigwedge_{p \in \mathrm{path}(s,v)} f_p(b_s)$$

ここで、$\mathrm{path}(s, v)$ は開始点 s から v への全ての経路からなる集合である。ここでも、不動点解のときと同様、「右辺が存在して、かつ x_v に等しい。」ということを表しているとする。

2.1.4 問題の例

いくつかの具体的な問題が上記の定式化の特殊な場合として特徴付けられる。

1. **グラフの到達可能性** L を $\{\bot, \top\}$ の2要素からなる束とする (もちろん $\bot \leq \top$)。ここで任意の $e \in E$ に対して f_e を恒等関数とし、b_s を $\bot$ とする。すると開始点 s から到達可能な $v \in V$ に対する解 x_v は $\bot$ となる。さらに、s から到達可能でない v に対しては、経路解 x_v は $\top$ になる。
2. **最短路問題** L を実数 $\mathbb{R}$ に最大値 $\top$ を付け加えた束とする。順序関係は $\mathbb{R}$ 上の通常の全順序

である。辺 e に関連づけられた距離を d_e とするとき、$f_e = \lambda x.\ x + d_e$ と定義する。$b_s = 0$ のとき、解 x_v は s から v への最短路に一致する。

3. **有限状態オートマトン** 辺 e のラベルがアルファベット Σ の要素 a_e になっているようなオートマトンを考える。L を Σ^* の全ての部分集合からなる集合とし、「$\supseteq$」による順序を入れる。s を初期状態と考え、b_s を空文字列 ϵ だけからなる集合とし、$f_e = \lambda x.\ \{wa_e \mid w \in x\}$ と定義する。すると、最終状態 v に対する経路解 x_v は、v で受理される文字列に対応する。
4. **データフロー解析** 「到達する定義」を求める問題を例にとって説明する。L をプログラム中の全ての定義の部分集合からなる集合とし、「$\subseteq$」による順序関係を入れる。プログラムの実行の様子はフローグラフに変換され、各辺 e には $f_e(x) = (x \cap K_e^c) \cup G_e$ で定義される関数 f_e が関連づけられる。ここで、K_e^c は e に対応する遷移で消滅する定義の集合の補集合であり、G_e は生成される定義の集合である。このとき、v における経路解 x_v は v に対応する実行ポイントで活きている定義の集合を表している。

2.1.5 反復解法アルゴリズム

文献 [14]では、不動点解を求めるアルゴリズムとして、図1のような反復解法とその正当性の証明 (ただし、計算機によって検証されたものではない) が示されている。

```
for each v ∈ V, x_v := ⊤;
x_{v_s} := b_s;
S := {v_s};
while S ≠ ∅ do
  S から一つ要素 v を取り除く;
  for each v --e--> w do
    if x_w ⋠ f_e(x_v) then
      x_w := x_w ∧ f_e(x_v);
      w を S に加える;
```

図1 抽象グラフ探索アルゴリズム

2.2 A* アルゴリズム

既に述べたように、2.1節の定式化は最短路問題を包括する。ここでは、最短路問題を解くための最適化手法の一つである A* アルゴリズム [12]について説明する。

A* アルゴリズムでは、各点 v に対して、ゴール地点までの距離の見積り $\mathrm{h}(v)$ が与えられている。A* アルゴリズムは、図1のアルゴリズムの

S から一つ要素 v を取り除く

の部分を、

「任意の $v' \in S$ について、$x_v + \mathrm{h}(v) \leq x_{v'} + \mathrm{h}(v')$」が成り立つような v を S から取り除く
もし v がゴールならば、アルゴリズムは終了する

と変更することで得られる。

もし $\mathrm{h}(v)$ が v からゴールまでの距離の下界を与えるならば、上記のアルゴリズムは (もし停止すれば) 最短路を与える。

2.3 HOL

形式的検証は証明検証系 HOL [7]を用いて行った。これは Church の「simple theory of types [5]」に基づく高階論理 (Higher Order Logic) の上で証明を行う検証系である。

実際の証明の作成は、意味的なまとまりをもった型・定数・定義・定理の集まりを作成することで行われ、これを theory と呼ぶ。HOL には既に整数やリストに関する theory が準備されており、ユーザはこれらの theory を使用して新しい theory を作っていく。使用する theory と使用されるとの間には依存関係が生じる。HOL を普通に起動した場合、「HOL」と呼ばれる theory の上に新しい theory を作るところから始まる。

theory には公理を与えることもできるが、これは一般に論理体系の無矛盾性を崩す可能性があるため、推奨される方法ではない。かわりに、定数を導入する際には必ずその定義を与える、という方針をとることにより、論理体系の無矛盾性を保持したまま定理の証明を進めていくという方法が用いられる。なお、定数を定義する場合には実際にその定数が何であるかの定義を与える方法 (constant definition) と、ある性質を満たす「もの」の存在性を示し、その「もの」(のうちの一つ) を定数の定義として指定する方法 (constant specification) がある。本

論文では、両者を単に「定義」と呼ぶことにする。

ここでHOLでの式に関する表記上の注意を記しておく。先頭に「⊢」があるものはそれが定理として証明されたことを示す。先頭に「$\vdash_{def}$」があるものはそれが定数の定義であることを示し、そこで定義されている定数には下線が引いてある。sans serifで書いてあるものは定数である。条件式は「if A then B else C」の形で表記され[†1]、Aはブール型、BとCは同じ型を持たなければならない。「*(** comments **)*」のようにイタリック体でコメントを入れることもある。

3 グラフ探索アルゴリズムの検証

3.1 全体的な構造

2.3節で述べたように、HOLでの検証はtheoryの作成によって行われ、theoryの間には依存関係が生じる。我々がグラフ探索アルゴリズムの検証で使用・作成したtheoryの依存関係は図2で示される通りになる。

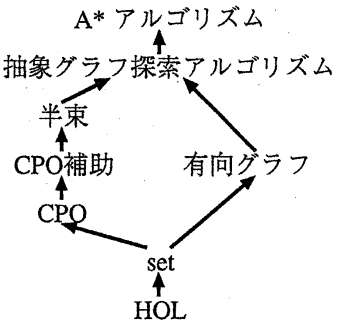

図2 theory間の依存関係

最下部にあるtheory「HOL」は、HOLでの検証の際にベースとなるtheoryであり、この下にさらに整数やリストなどに関するtheoryがある。以下、図2中のそれぞれのtheoryについて簡単に説明する。

3.2 CPOと半束

順序関係に関するtheoryとして、CamilleriとPrasetyaによるCPO(Complete Partial Orders) theory [3]の一部と、さらに補助のtheoryを作成して利用した。半束上の操作の多くはCPOのそれを流用している。

- Cap$r\,x\,y$：順序rに関するxとyの最大下界。
- Top$r$ ： A上の順序rに関するA中の最大要素。
- Mono$r\,s\,f$ ： 関数$f : A \to B$がA上の順序rとB上の順序sについて単調であることを示す。すなわち、
$$\forall x\,y.\ r\,x\,y \Longrightarrow s\,(f\,x)(f\,y)$$
- Distrib$r\,s\,f$ ： 関数$f : A \to B$がA上の順序rとB上の順序sについて分配的であることを示す。すなわち、
$$\forall x\,y.\ f\,((\mathsf{Cap}\,r)\,x\,y) = (\mathsf{Cap}\,s)\,(f\,x)\,(f\,y)$$

最後のもの以外はCPO theoryによる。

半束lは「$\wedge$」に対応する操作meetから定義されるのだが、この操作は形式的検証ではあまり用いられない。かわりに「$\wedge$」から導入される順序関係LEQが中心的な役割を果たす。このことは上記のCPO theory中でほとんどの操作が順序関係を引数に取っていることからもわかる。

$$\vdash_{def} \underline{\mathsf{LEQ}}\,l\,x\,y \Longleftrightarrow (\mathsf{meet}\,l\,x\,y = x)$$

結果として、半束上の操作は「Cap(LEQ l) $x\,y$」や「Top(LEQ l)」などのように、少し冗長な形で表記される。

3.3 有向グラフ

有向グラフの基本的な定義は、Wong [15]に従った。有向グラフ上の主要な操作は以下の通り。

- VS$\,g$：グラフgの頂点の集合。
- ES$\,g$：グラフgの辺の集合。
- es$\,e$： 辺eの始点(前節で$h(e)$と書かれていたもの)。
- ed$\,e$： 辺eの終点(前節で$t(e)$と書かれていたもの)。
- INCIDENT_FROM$\,g\,v$：グラフgで点vから出ていく辺の集合(前節でout(v)と書かれていたもの)。
- EMPTY_PATH$\,v$ ： vから始まる空の経路$(v, [\,])$。
- PATH_SNOC$\,p\,e$：辺eを$p = (v, [e_1, \ldots, e_n])$の最後に付け加えてできる経路$(v, [e_1, \ldots, e_n, e])$。

†1 なお、実際のHOLの表記では条件式は「$A \Rightarrow B \mid C$」と表される。

3.4 抽象グラフ探索アルゴリズム

抽象グラフ探索アルゴリズムは次の形の述語として定義される。

POTENTIAL $n\ g\ l\ s\ bs\ fe\ open\ closed\ outv\ xv$.

述語 POTENTIAL の引数は以下の3種類に分類される。

1. 実行カウンタとなるもの。
   引数 n がこの分類に属する。この変数は現在何ステップ目を実行しているかを記録する。また、この引数が存在することによって、述語 POTENTIAL を定義する際に自然数に関する再帰的定義が使用できるようになっている。
2. 問題の仕様として与えられるもの。
   これらの引数は実行中は変化しない。g, l, s, bs, fe がこの分類に属する。これらの意味は以下の通りである。
   - g：有向グラフ。
   - l：半束。
   - s：探索の開始点。
   - bs：s における初期値。
   - fe ： 各辺における関数の割り当て。「$fe\,e\,x$」が「$f_e(x)$)」に対応する。
3. 実行中の内部状態を表すもの。
   これらは実行が1ステップ進むに従って変化する。$open$, $closed$, $outv$, xv がこの分類に属する。これらの意味は以下の通りである。
   - $open$：未処理の点の集合 (図1での S に対応する)
   - $closed$：少なくとも一度処理された点で、$open$ に入っていないもの
   - $outv$：展開される点から出ていく辺で、まだ処理していないものの集合
   - xv ： 各点に対する l の要素の割り当て。「$xv\,v$」が x_v) に対応する。

述語 POTENTIAL は図3のように自然数に関する再帰的定義によって定義される。n が0の場合が初期状態 (0ステップ目) で満たすべき性質を表し、(SUC n) ステップ目 (n ステップ目の次) のときの状態は n ステップ目のときの状態との関係で書き表わされている。このアルゴリズムは $open$ と $outv$ の両方が空集合になったときに終了する。ここでは、終了状態になった後は同じ状態を繰り返すように定義してある。

```
⊢def
(∀g l s bs fe open closed outv xv.
 POTENTIAL 0 g l s bs fe open closed outv xv ⟺
 (open = {s}) ∧ (closed = {}) ∧ (outv = {}) ∧
 (xv = (λv. if (v = s) then bs else (Top(LEQ l))))) ∧
(∀n g l s bs fe open closed outv xv.
 POTENTIAL(SUC n) g l s bs fe open closed outv xv ⟺
 (∃open' closed' outv' xv'.
  POTENTIAL n g l s bs fe open' closed' outv' xv' ∧
  (if (outv' = {})
   then
   (if (open' = {})
    then ((open = open') ∧ (closed = closed') ∧
          (outv = outv') ∧ (xv = xv'))
    else (∃v. v ∈ open' ∧ (closed = {v} ∪ closed') ∧
          (open = open' \ {v}) ∧
          (outv = INCIDENT_FROM g v) ∧ (xv = xv')))
   else (** (outv' ≠ {}) **)
   (∃e. e ∈ outv' ∧ (outv = outv' \ {e}) ∧
    (if (LEQ l (xv' (ed e)) (fe e (xv' (es e))))
     then ((open = open') ∧ (closed = closed') ∧
           (xv = xv'))
     else ((open = {ed e} ∪ open') ∧
           (closed = closed' \ {ed e}) ∧
           (xv = (λv. if (v = ed e)
                      then (Cap(LEQ l) (xv'(ed e))
                                       (fe e (xv' (es e))))
                      else (xv' v)))))))))
```

図3 抽象グラフ探索アルゴリズムの定義

もとのアルゴリズムは2重ループになっていたのに対し、形式化されたアルゴリズムは1重のループになっていることに注意。もとの内側のループは外側のループに吸収される形になっており、直前のステップの $outv'$ の値によってもとの内側のループか外側のループかを判定する。$outv' = \{\}$ であればもとの外側のループである。

この定義を用いて、アルゴリズムが停止したときの解の正当性を保証する以下の定理を証明した。

1. アルゴリズム POTENTIAL は、もし停止すれば、不動点解を与える。

```
⊢ ∀g l s bs fe. s ∈ VS g ∧ (∀e. ed e ≠ s) ∧
   HAS_Top l ∧ (bs ≠ Top(LEQ l)) ∧
   (∀x e. e ∈ ES g ∧ (fe e x = Top(LEQ l)) ⟹
     (x = Top(LEQ l))) ∧
   (∀e. e ∈ ES g ⟹ Mono(LEQ l) (LEQ l) (fe e)) ⟹
    (∀n closed xv.
     POTENTIAL n g l s bs fe {} closed {} xv ⟹
     IS_FPSOLN l g fe bs s xv)
```

2. アルゴリズム POTENTIAL は、もし停止すれば、経路解を与える。

$$\begin{array}{l}\vdash \forall g\ l\ s\ bs\ fe.\ s \in \mathsf{VS}\,g \wedge (\forall e.\ \mathsf{ed}\,e \neq s) \wedge \\ \quad \mathsf{HAS\_Top}\,l \wedge (bs \neq \mathsf{Top}(\mathsf{LEQ}\,l)) \wedge \\ \quad (\forall x\ e.\ e \in \mathsf{ES}\,g \wedge (fe\ e\ x = \mathsf{Top}(\mathsf{LEQ}\,l)) \Longrightarrow \\ \qquad (x = \mathsf{Top}(\mathsf{LEQ}\,l))) \wedge \\ \quad (\forall e.\ e \in \mathsf{ES}\,g \Longrightarrow \mathsf{Distrib}(\mathsf{LEQ}\,l)\ (\mathsf{LEQ}\,l)\ (fe\ e)) \Longrightarrow \\ \qquad (\forall n\ closed\ xv. \\ \qquad\quad \mathsf{POTENTIAL}\,n\ g\ l\ s\ bs\ fe\ \{\}\ closed\ \{\}\ xv \Longrightarrow \\ \qquad\quad \forall v.\ v \in \mathsf{VS}\,g \Longrightarrow \\ \qquad\qquad \mathsf{IS\_PATHSOLN}\,l\ g\ fe\ bs\ s\ v\ (xv\ v)))\end{array}$$

述語 IS_FPSOLN, IS_PATHSOLN はその最後の引数がそれぞれ不動点解、経路解であることを表している。不動点解については fe が単調であることのみで十分であるが、経路解については fe が分配的であることが要求される。

不動点解の正当性に関する定理の証明には 13 個の中間的な補題 (そのうちの 2 つは 4節で述べる) を要し、経路解の正当性にはさらに 1 つの定義と 8 つの補題を要した。これらの補題のほとんどはステップ数を表す自然数 n に関する帰納法で証明できた。これらの定理を証明するのに要した HOL のスクリプトは全体で約 1300 行である。

4 A* への応用

この節では、A* アルゴリズムの定義と、それに関して示した定理について述べる。通常 A* アルゴリズムは $\mathbb{R} \cup \{\top\}$ の上で定義されるが、ここでは一般の半束を用いて定義する。

2.2節で述べたように、A* アルゴリズムは抽象グラフ探索アルゴリズムの拡張として表現できるため、その定義は抽象グラフ探索アルゴリズムの定義と同じような形になる (図 4)。

定義中で、$\mathsf{IS\_MINIMAL}\,r\,X\,x$ は $x \in X$ が集合 X の要素中で順序関係 r について極小であることを示す。

前節の POTENTIAL の引数に比べ、ASTAR には 4 つの引数が加わっている。$goal$ と hv は問題の仕様として与えられるもので、$stat$ と po は内部状態である。これらの引数の意味は以下の通り。

- $goal$: ゴールの点。
- hv : 各点に対するゴールまでの距離の見積りを表す関数の割り当て。
- $stat$: 実行ステータス。
- po : 開始点 s ら各点への (それまでの) 最適パス

$$\begin{array}{l}\vdash_{def}\ (\forall g\ l\ s\ bs\ fe\ goal\ hv\ open\ closed\ outv\ xv\ stat\ po. \\ \quad \underline{\mathsf{ASTAR}}\ 0\ g\ l\ s\ bs\ fe\ goal\ hv\ open\ closed\ outv\ xv\ stat\ po \Longleftrightarrow \\ \quad (open = \{s\}) \wedge (closed = \{\}) \wedge (outv = \{\}) \wedge \\ \quad (xv = (\lambda v.\ \ \mathtt{if}\ (v = s)\ \mathtt{then}\ bs\ \mathtt{else}\ (\mathsf{Top}(\mathsf{LEQ}\,l)))))) \wedge \\ \quad (stat = 0) \wedge \\ \quad (po = \lambda v.\ \ \mathsf{EMPTY\_PATH}\,s)) \wedge \\ (\forall n\ g\ l\ s\ bs\ fe\ goal\ hv\ open\ closed\ outv\ xv\ stat\ po. \\ \quad \underline{\mathsf{ASTAR}}(\mathsf{SUC}\,n)\ g\ l\ s\ bs\ fe\ goal\ hv\ open\ closed\ outv\ xv\ stat\ po \Longleftrightarrow \\ \quad \exists open'\ closed'\ xv'\ outv'\ stat'\ po'. \\ \quad \mathsf{ASTAR}\,n\ g\ l\ s\ bs\ fe\ goal\ hv\ open'\ closed'\ outv'\ xv'\ stat'\ po' \wedge \\ \quad (\mathtt{if}\ (stat' = 0) \\ \qquad \mathtt{then} \\ \qquad (\mathtt{if}\ (outv' = \{\}) \\ \qquad\quad \mathtt{then} \\ \qquad\quad (\mathtt{if}\ (open' = \{\}) \\ \qquad\qquad \mathtt{then}\ ((open = open') \wedge (closed = closed') \wedge \\ \qquad\qquad\qquad\quad (outv = outv') \wedge \\ \qquad\qquad\qquad\quad (xv = xv')) \wedge (stat = 2) \wedge (po = po')) \\ \qquad\qquad \mathtt{else}\ (\exists v.\ v \in open' \wedge \\ \qquad\qquad\qquad\quad (\mathsf{IS\_MINIMAL}(\mathsf{LEQ}\,l)\ \{hv\ v_1\ (xv'\ v_1) \mid v_1 \in open'\} \\ \qquad\qquad\qquad\qquad (hv\ v\ (xv'\ v))) \wedge \\ \qquad\qquad\qquad\quad (\mathtt{if}\ (v = goal) \\ \qquad\qquad\qquad\qquad \mathtt{then}\ ((stat = 1) \wedge (open = open') \wedge \\ \qquad\qquad\qquad\qquad\qquad\quad (closed = closed') \wedge \\ \qquad\qquad\qquad\qquad\qquad\quad (outv = outv') \wedge (xv = xv') \wedge (po = po')) \\ \qquad\qquad\qquad\qquad \mathtt{else}\ ((stat = stat') \wedge (closed = \{v\} \cup closed') \wedge \\ \qquad\qquad\qquad\qquad\qquad\quad (open = open' \setminus \{v\}) \wedge \\ \qquad\qquad\qquad\qquad\qquad\quad (outv = \mathsf{INCIDENT\_FROM}\,g\,v) \wedge \\ \qquad\qquad\qquad\qquad\qquad\quad (xv = xv') \wedge (po = po'))) \\ \qquad\quad \mathtt{else}\ (**\ (outv' \neq \{\})\ **) \\ \qquad\quad (\exists e.\ e \in outv' \wedge (outv = outv' \setminus \{e\}) \wedge (stat = stat') \wedge \\ \qquad\quad\ (\mathtt{if}\ (\mathsf{LEQ}\,l\ (xv'\ (\mathsf{ed}\,e))\ (fe\ e\ (xv'\ (\mathsf{es}\,e)))) \\ \qquad\quad\qquad \mathtt{then}\ ((open = open') \wedge (closed = closed') \wedge \\ \qquad\quad\qquad\qquad\quad (xv = xv') \wedge (po = po') \\ \qquad\quad\qquad \mathtt{else}\ ((open = \{\mathsf{ed}\,e\} \cup open') \wedge \\ \qquad\quad\qquad\qquad\quad (closed = closed' \setminus \{\mathsf{ed}\,e\}) \wedge \\ \qquad\quad\qquad\qquad\quad (xv = (\lambda v.\ \ \mathtt{if}\ (v = \mathsf{ed}\,e) \\ \qquad\quad\qquad\qquad\qquad\qquad\quad \mathtt{then}\ (\mathsf{Cap}(\mathsf{LEQ}\,l)\ (xv'(\mathsf{ed}\,e)) \\ \qquad\quad\qquad\qquad\qquad\qquad\qquad\quad (fe\ e\ (xv'\ (\mathsf{es}\,e)))) \\ \qquad\quad\qquad\qquad\qquad\qquad\quad \mathtt{else}\ (xv'\ v)) \wedge \\ \qquad\quad\qquad\qquad\quad (po = (\lambda v.\ \ \mathtt{if}\ (v = \mathsf{ed}\,e) \\ \qquad\quad\qquad\qquad\qquad\qquad\quad \mathtt{then}\ \mathsf{PATH\_SNOC}(po'\ (\mathsf{es}\,e))\,e \\ \qquad\quad\qquad\qquad\qquad\qquad\quad \mathtt{else}\ po'\ v)))))) \wedge \\ \qquad \mathtt{else}\ (**\ (stat \neq 0)\ **) \\ \qquad ((open = open') \wedge (closed = closed') \wedge (outv = outv') \wedge \\ \qquad\ (xv = xv') \wedge (stat = stat') \wedge (po = po'))))\end{array}$$

図 4 A* Algorithm の定義

の割り当て。

hv と *stat* については説明を要するであろう。2.2節では、点 v に対する見積り $\mathrm{h}(v)$ は実数であり、S から点 v を取り出すときの基準として $(x_v + \mathrm{h}(v))$ が用いられていた。しかし図4の形式化では半束 l を $\mathbb{R} \cup \{\top\}$ に制限するのではなく、$(hv\,v)$ を l から l への関数とし、v を取り出すときの基準として $(hv\,v\,(xv\,v))$ を用いるようにしている。実際、この定式化は実数の場合の拡張になっている $(hv\,v = \lambda x.\ x + \mathrm{h}(v)$ ととる)。実行ステータス *stat* は 0, 1, 2 のいずれかの値をとる。0 は探索を実行中であるということ、1 は探索がゴールの点まで到達し、実行が終了したことを表す。2 はゴールの点に到達しないまま実行が終了したことを表す (*goal* が開始点から到達可能でない場合にこの状態になる)。$stat = 0$ である間は内部状態が POTENTIAL の場合と同様に変化し、*stat* が 1 か 2 になると内部状態は変化しなくなる。

2.2節で述べた通り、ASTAR と POTENTIAL の決定的な違いは、集合 *open* からどのように点を選んでくるかである。ここでは、半束 l 中の順序関係「LEQ l」に関して $(hv\,v\,(xv\,v))$ 極小を極小にする $v(\in open)$ を選ぶようにしている。(LEQ l が全順序であることは仮定していないので、最小を達成する点が存在するとは限らない。) このことについては、5.2節で詳しく触れることとする。

POTENTIAL に関するほとんどの性質は、以下の定理を用いて ASTAR に関する性質として利用することができる。

$\vdash$ $\forall g\,l\,n\,s\,bs\,fe\,goal\,hv\,open\,closed\,outv\,xv\,stat\,po.$
(ASTAR $n\,g\,l\,s\,bs\,fe\,goal\,hv\,open\,closed\,outv\,xv\,stat\,po \Longrightarrow$
($\exists m.$ (POTENTIAL $m\,g\,l\,s\,bs\,fe\,open\,closed\,outv\,xv$)))

上記の定理自身は、ASTAR と POTENTIAL をその定義で展開し、n に関する帰納法を用いることで証明できる。この定理により、

$$\forall g\,l \ldots.\ \mathsf{POTENTIAL} \ldots \Longrightarrow P$$

の形の定理から

$$\forall g\,l \ldots.\ \mathsf{ASTAR} \ldots \Longrightarrow P$$

の形の定理をただちに得ることができる。

POTENTIAL の正当性を示すのに用いられた以下の2つの定理は、ASTAR の正当性の証明でも重要な役割を果たす。これらは文献 [14]の Assertion 1 と Assertion 3 に対応する。

- 任意の辺 $e = (v, w)$ について、もし e が既に処理されていれば、$x_w \leq f_e(x_v)$ である。

$\vdash$ $\forall g\,l\,n\,s\,bs\,fe\,open\,closed\,outv\,xv.$
POTENTIAL $n\,g\,l\,s\,bs\,fe\,open\,closed\,outv\,xv$
$\Longrightarrow \forall e.\ e \in \mathsf{ES}\,g \wedge \mathsf{es}\,e \in closed \wedge e \notin outv$
$\Longrightarrow$ (LEQ l) (xv (ed e)) ($fe\,e$ (xv (es e)))

- 任意の辺 $e = (v, w)$(ただし $v \in closed$) について、$w \in open \cup closed$ であるかまたは e が処理済みである。

$\vdash$ $\forall g\,l\,s\,bs\,fe.$ HAS_Top $l \wedge (bs \neq \mathsf{Top}(\mathsf{LEQ}\,l)) \wedge$
($\forall x\,e.\ e \in \mathsf{ES}\,g \wedge (fe\,e\,x = \mathsf{Top}(\mathsf{LEQ}\,l)) \Longrightarrow$
$(x = \mathsf{Top}(\mathsf{LEQ}\,l))) \Longrightarrow$
$\forall n\,open\,closed\,outv\,xv.$
POTENTIAL $n\,g\,l\,s\,bs\,fe\,open\,closed\,outv\,xv$
$\Longrightarrow \forall e.\ e \in \mathsf{ES}\,g \wedge \mathsf{es}\,e \in closed$
$\Longrightarrow \mathsf{ed}\,e \in open \vee \mathsf{ed}\,e \in closed \vee e \in outv$

POTENTIAL に関する上記の定理を ASTAR に対して適用することにより、以下の補題が得られる。

$\vdash$ $\forall g\,l\,n\,s\,bs\,fe\,goal\,hv\,open\,closed\,outv\,xv\,po.$
ASTAR $n\,g\,l\,s\,bs\,fe\,goal\,hv\,open\,closed\,outv\,xv\,1\,po \wedge$
HAS_Top $l \wedge (bs \neq \mathsf{Top}(\mathsf{LEQ}\,l)) \wedge$
($\forall x\,e.\ e \in \mathsf{ES}\,g \wedge (fe\,e\,x = \mathsf{Top}(\mathsf{LEQ}\,l)) \Longrightarrow$
$(x = \mathsf{Top}(\mathsf{LEQ}\,l))) \wedge$
($\forall e.\ e \in \mathsf{ES}\,g \Longrightarrow \mathsf{Mono}(\mathsf{LEQ}\,l)\,(\mathsf{LEQ}\,l)\,(fe\,e)) \wedge$
($\forall bv\,v.$ IS_LEQ_PATHSOLN $l\,g\,fe\,(hv\,v\,bv)\,bv\,v\,goal) \wedge$
($\forall v.\ (v \in open) \Longrightarrow (\mathsf{LEQ}\,l)\,(xv\,goal)\,(hv\,v\,(xv\,v))$)
$\Longrightarrow$ IS_LEQ_PATHSOLN $l\,g\,fe\,(xv\,goal)\,bs\,s\,goal$

ここで IS_LEQ_PATHSOLN $l\,g\,fe\,x\,b\,v\,w$ は x が

$$\{((fe\,e_n) \circ \cdots \circ (fe\,e_1))(b) \mid (v, [e_1, \ldots, e_n])\text{が}v\text{から}w\text{への経路}\}$$

の下界であるということを表している。

上記の補題の条件部分で IS_LEQ_PATHSOLN を使用している部分は、「$\mathrm{h}(v)$ が v からゴールへの下界である」という条件に対応する。

A* アルゴリズムに関して示したメインの定理は、それが正常終了 $(stat = 1)$ したときに経路解を与えるというものである。経路解はある集合の最大下界で表されるので、

- 解がある集合の下界であること。
- 解がそのような下界の中で最大であること。

の2つを示さなければならない。最初の性質は上の補題によって得られる。2つめの性質は「LEQ l」が全順序であるという条件を付加して、$(po\,goal)$ が実際に s から *goal* までの経路を記録し、その距離が $(xv\,goal)$ と一致することを示すことによって示される。最終的に、次の定理が得られた。

$\vdash$ $\forall g\, l\, n\, s\, bs\, fe\, goal\, hv\, open\, closed\, outv\, xv\, po.$
ASTAR $n\, g\, l\, s\, bs\, fe\, goal\, hv\, open\, closed\, outv\, xv\, 1\, po \wedge$
HAS_Top $l \wedge (bs \neq \mathsf{Top}(\mathsf{LEQ}\, l)) \wedge$
$(\forall x\, e.\ e \in \mathsf{ES}\, g \wedge (fe\, e\, x = \mathsf{Top}(\mathsf{LEQ}\, l)) \Longrightarrow$
$(x = \mathsf{Top}(\mathsf{LEQ}\, l))) \wedge$
$(\forall e.\ e \in \mathsf{ES}\, g \Longrightarrow \mathsf{Mono}(\mathsf{LEQ}\, l)\, (\mathsf{LEQ}\, l)\, (fe\, e)) \wedge$
$(\forall bv\, v.$ IS_LEQ_PATHSOLN $l\, g\, fe\, (hv\, v\, bv)\, bv\, v\, goal) \wedge$
$(\forall b.\ hv\, goal\, b = b) \wedge$ (IS_TOTAL l)
$\Longrightarrow$ IS_PATHSOLN $l\, g\, fe\, bs\, s\, goal\, (xv\, goal)$

5 形式的検証による知見

この節では形式的検証によって得られた知見について述べる。

5.1 もとの証明に対する変更

まず、開始点 s からの到達可能性を考慮に入れるように不動点解の定義を変更した。2.1.2節では到達可能性を考慮に入れるようにしてあるが、文献[14]でのもとの不動点解は以下のようなものであった。

$$\begin{cases} x_s &= b_s \\ x_v &= \bigwedge_{t(e)=v} f_e(x_{h(e)}) \quad (v \neq s) \end{cases}$$

しかし、このままでは抽象グラフ探索アルゴリズム(図1)が正しい解を求められない場合がある。例えば、孤立した点(その点に入ってくる辺が存在しない点)に対しては $\top$ 以外の値を割り当てることができてしまう。

また、文献[14]では有向グラフは有限なものに限られていた。以下の形の定理を示すために、束 L の任意の最大下界の存在を使わなければならなかったためである。

> 任意の実行ステップ n において、
> a が集合 $X(n)(\subseteq L)$ の最大下界であれば、
> $P(a)$ が成り立つ。

もし上記の命題を n に関する帰納法で示そうとすると、以下の仮定から「$P(a)$ が成り立つ」ことを導き出さなければならない。

- (仮定1) 任意の実行ステップ n において、a' が集合 $X(n)$ の最大下界であれば、$P(a')$ が成り立つ。
- (仮定2) a は集合 $X(n+1)$ の下界である。

しかし、(仮定1)からは何も導きだすことができない。なぜならば、L の完全性かグラフの有限性を用いないかぎり、$X(n)$ の最大下界の存在性は言えず、よって(仮定1)の条件部分を消すことができないためである。

そこで、我々は上記の命題を以下のようにより一般的な命題に変更した。

> 任意の実行ステップ n において、
> a が集合 $X(n)(\subseteq L)$ の下界であれば、
> $P(a)$ が成り立つ。

$X(n)$ の最大下界が存在するという条件の下では、上記の命題が何ら一般性を失っていないことに注意。同様に帰納法を用いると、以下の仮定から「$P(a)$ が成り立つ」ことを証明すればよいことになる。

- (仮定1') 任意の実行ステップ n において、a' が集合 $X(n)$ の下界であれば、$P(a')$ が成り立つ。
- (仮定2') a は集合 $X(n+1)$ の下界である。

実は「$X(n+1)$ の任意の下界は、$X(n)$ の下界でもある」ということは簡単に示せるので、この場合は(仮定1')が利用できる。よって、変更後はグラフの有限性を仮定することなく正当性を証明できた。

5.2 A* アルゴリズム

4節ではA* アルゴリズムに関する補題を述語 ASTAR に関する性質として述べた。しかし、その証明を注意深く検討してみると、述語 ASTAR に関する性質は証明中で使用されていないことに気付いた。実際、以下のようにA* アルゴリズムに関する性質を述語 POTENTIAL を用いて示すことが出来る。

$\vdash$ $\forall g\, l\, n\, s\, bs\, fe\, goal\, hv\, open\, closed\, outv\, xv.$
POTENTIAL $n\, g\, l\, s\, bs\, fe\, open\, closed\, outv\, xv \wedge$
HAS_Top $l \wedge (bs \neq \mathsf{Top}(\mathsf{LEQ}\, l)) \wedge$
$(\forall x\, e.\ e \in \mathsf{ES}\, g \wedge (fe\, e\, x = \mathsf{Top}(\mathsf{LEQ}\, l)) \Longrightarrow$
$(x = \mathsf{Top}(\mathsf{LEQ}\, l))) \wedge$
$(\forall e.\ e \in \mathsf{ES}\, g \Longrightarrow \mathsf{Mono}(\mathsf{LEQ}\, l)\, (\mathsf{LEQ}\, l)\, (fe\, e)) \wedge$
$(\forall bv\, v.$ IS_LEQ_PATHSOLN $l\, g\, fe\, (hv\, v\, bv)\, bv\, v\, goal) \wedge$
$(outv = \{\}) \wedge (goal \notin closed) \wedge$
$(\forall v.\ (v \in open) \Longrightarrow (\mathsf{LEQ}\, l)\, (xv\, goal)\, (hv\, v\, (xv\, v)))$
$\Longrightarrow$ IS_LEQ_PATHSOLN $l\, g\, fe\, (xv\, goal)\, bs\, s\, goal$

A* アルゴリズムの定義(図4)では、ある指標によって極小となる点を集合 *open* から取り除くようになっていた。しかし、上記の定理は「取り除く点をどのように選ぶか」はA* アルゴリズムの**正当性**に関しては**無関係**であることを物語っている。(実際は点の選び方は効率、すなわち、探索空間のサイズには影響を与えている。) 正当性において重要なのは点の選び方ではなく、最終ステージ(アルゴリ

ズムが正常終了する一歩手前) で
$\forall v.\ (v \in open) \Longrightarrow (\mathsf{LEQ}\, l)\,(xv\ goal)\,(hv\,v\,(xv\,v))$
という性質が成り立っているかどうかである。この条件は任意の $v \in open$ に対して、ゴールの点に割り当てられている値が、$(hv\,v\,(xv\,v))$ 以下であるということを示している。なお、通常の $\mathbb{R} \cup \{\top\}$ 上の A* アルゴリズムでは ($(\mathsf{LEQ}\, l)$ が全順序であるという仮定のもとで) 最終ステージで上記の条件は成り立つようになっている。

この事実は「$open$ の中で $x_v + \mathrm{h}(v)$ が最小になるような点 v を必ずしも選ぶわけではない」という、A* アルゴリズムの変種の存在の可能性を示している。例えば、$open$ から点を選ぶときに無限ループを避けるような方針で選ぶようにすれば、通常の A* アルゴリズムでは解けない (終了しない) 問題でも解けるようになる可能性がある。示すべきことは最終ステージでの条件であって、途中状態での点の選び方は正当性には関係無いからである。

4節では、アルゴリズムによって得られた解がある集合の下界のうち最大であることを示すために、「$\mathsf{LEQ}\, l$」が全順序であるという仮定を置いた。しかし、これは全順序に関する条件を置かなくても示すことができる (ただし「$\mathsf{LEQ}\, l$」の分配性は必要)。このことに関する定理もやはり POTENTIAL の性質として示すことができ、POTENTIAL と ASTAR の両方の正当性の証明に利用することができる。†2

$$
\begin{aligned}
\vdash\ & \forall g\, l\, s\, bs\, fe.\ s \in \mathsf{VS}\, g \wedge \mathsf{HAS\_Top}\, l \wedge \\
& (\forall e.\ e \in \mathsf{ES}\, g \Longrightarrow \mathsf{Distrib}(\mathsf{LEQ}\, l)\,(\mathsf{LEQ}\, l)\,(fe\, e)) \\
& \Longrightarrow \forall n\ open\ closed\ outv\ xv. \\
& \mathsf{POTENTIAL}\, n\, g\, l\, s\, bs\, fe\ open\ closed\ outv\ xv \\
& \Longrightarrow \forall v\, a.\ \mathsf{IS\_LEQ\_PATHSOLN}\, l\, g\, fe\, a\, bs\, s\, v \\
& \Longrightarrow (\mathsf{LEQ}\, l)\, a\, (xv\, v)
\end{aligned}
$$

6 抽象到達可能性検査

前節までに、抽象グラフ探索アルゴリズムに関する形式的検証と、その検証結果の、具体的な最適化アルゴリズム (A* アルゴリズム) の検証への利用について述べた。しかし、抽象グラフ探索アルゴリズムは確かに様々なグラフアルゴリズムを包括しているものの、モデル検査に用いられるアルゴリズムの検証には向いていない。そこで、本節ではモデル検査アルゴリズムの検証を視野に入れた別の抽象アルゴリズムについて述べる。

†2 この考察は当時我々の研究室を訪れていた ENS Lyon の Bruno Martin [11]によって行われた。

6.1 ラベル付き遷移システム

検査対象のシステムは、ラベル付き遷移システムで記述される。ラベル付き遷移システムは状態の集合 $\mathcal{S}$ とアクションの集合 $\mathcal{A}$、遷移関係 $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$、初期状態の集合 $\mathcal{I}$ から成る。$(s, a, t) \in \mathcal{R}$ のことを $s \xrightarrow{a} t$ と書く。ラベル付き遷移システムは状態を頂点、遷移関係を辺と考えると、これまで扱ってきた有向グラフになる。以下では、有向グラフのときと同様に初期状態が 1 つのときを考える。

6.2 到達可能性検査

システムの安全性を検証するモデル検査法の一つに、グラフの到達可能性を用いるものがある。この方法では、システムの安全性を時相論理の論理式で表現し、この論理式の否定をオートマトンによって表現してシステムとの積をとり、危険な状態が初期状態から到達可能であるかどうかを判定することによって安全性の検査を行う。

グラフの到達可能性判定アルゴリズムは図 5のように表現される。

```
1:  S := {s0};
2:  U := S;
3:  while S ≠ ∅ do
4:      S から一つ要素 s を取り除く;
5:      for each s -a-> t do
6:          if t ∉ U then
7:              t を S と U に加える;
```

図 5 到達可能性検査アルゴリズム

s_0 は初期状態、S は抽象グラフ探索アルゴリズムのときと同様、未処理の状態からなる集合である。U はそれまでの検索で初期状態から到達可能と判断された状態の集合となり、アルゴリズム終了時には初期状態から到達可能な状態全ての集合となる。

ここで、状態間に順序関係 $\leq$ ($\subseteq \mathcal{S} \times \mathcal{S}$) を導入する。$s \leq s'$ は、状態 s' が状態 s よりもより一般的 (あるいは抽象的) な状態であるということを意味する。より抽象度の高い状態で複数 (あるいは無限) の状態を代表させることにより、探索空間を縮小させたり、本来有限の状態空間で使用されるモデル検査アルゴリズムを、「無限の状態空間を有限の探索空間で網羅」することにより、無限の状態空間の場合

にも適用させるのがねらいである。

状態の間に順序を入れた場合は、6行目以降が変化する (図6)。

```
 1:  S := {s_0};
 2:  U := S;
 3:  while S ≠ ∅ do
 4:    S から一つ要素 s を取り除く;
 5:    for each s -a-> t do
 6:      if ∃t_C ∈ U. t_C ≥ t then (* Cover *)
 7:        (* 何もしない *);
 8:      else if C_G then (* Generate *)
 9:        t_G ∉ U, t ≤ t_G を満たす t_G を作る;
10:        t' ≤ t_G となる t' ∈ U を
           U と S からいくつか取り除く;
11:        t_G を S と U に加える;
12:      else (* Step *)
13:        t' ≤ t となる t' ∈ U を
           U と S からいくつか取り除く;
14:        t を S と U に加える;
```

図6　抽象到達可能性検査アルゴリズム

図5は図6で状態間の順序を $s \leq s' \stackrel{\text{def}}{\Longleftrightarrow} (s = s')$ とし、$C_G \stackrel{\text{def}}{=} \textbf{false}$ とした場合に相当する。

図6のアルゴリズムはあくまで個々の具体的なアルゴリズムを表現するための枠組であり、C_G や t_G は具体的なアルゴリズムによって指定される。抽象アルゴリズムに関する性質を証明する際には、示したい性質について C_G や t_G に関する条件を抽象的に表してそれを用いる。個々の具体的なアルゴリズムについては、「C_G や t_G に関する条件が満たされる」ということを示すことによって検証を行うのである。

この抽象到達可能性検査アルゴリズムでは、以下のことが成り立つ。

性質1 ラベル付き遷移システム上の遷移列

$$s_0 \stackrel{a_0}{\to} s_1 \stackrel{a_1}{\to} \cdots \stackrel{a_{n-1}}{\to} s_n$$

について、「$\forall s'_i.\ s_i \leq s'_i \Longrightarrow \exists s'_{i+1}.\ s'_i \stackrel{a_i}{\to} s'_{i+1}$ かつ $s_{i+1} \leq s'_{i+1}$」が成り立つならば、アルゴリズム終了時に $s'_n \in U$ となる s'_n が存在して、$s_n \leq s'_n$ が成立する。

7　抽象到達可能性検査の例

7.1　Covering Graph Construction

文献 [6]で Emerson らは無限個の状態を持つシステムのモデル検査のアルゴリズムのいくつかが、"Covering Graph" を作成する到達可能性検査で記述されることを示した。前節の抽象到達可能性検査アルゴリズムは、Covering Graph 作成のアルゴリズムを基に、アルゴリズム自体の検証を念頭に入れて抽象化したものである。

Emerson らの Covering Graph 作成では、具体的な遷移システムでの状態と、Covering Graph により表される抽象的なシステムでの状態が分離されていたが、我々のアルゴリズムではそれらを区別せず、抽象的な状態の一部を具体的な状態とみなす。また、文献 [6]ではグラフの頂点 n とその上のラベルとしての uniform subset $\mathsf{L}(n)$ を別々に扱っているが、ここではそれらを同一視する (uniform subset 自体が状態だと考える)。さらに、Covering Graph 作成は文字通り「グラフの作成」であるが、ここではグラフは作成せず、「作成されるグラフに現れる状態の探索」を行うものとする。

ここでは Covering Graph 作成について細かく述べず、Covering Graph 作成と抽象到達性検査の間のおおまかな対応を述べることにする。Covering Graph 作成の詳細については文献 [6]を参照されたい。

- ラベル付き遷移システム

$$\begin{cases} S \stackrel{\text{def}}{=} \{rep(U) \mid U \text{は uniform subset}\} \\ \qquad (\text{抽象状態の representative の集合}) \\ \mathcal{A} \stackrel{\text{def}}{=} \text{アクションの集合} \\ s \stackrel{a}{\to} t \stackrel{\text{def}}{\Longleftrightarrow} t = rep(\bar{a}(s)) \\ \qquad (\bar{a}(s) \text{は} \{q \mid p \stackrel{a}{\to} q, p \in s\}) \\ \mathcal{I} \stackrel{\text{def}}{=} \{rep(X) \mid X \text{は初期状態を} \\ \qquad \text{uniform subset に分割したもの}\} \\ s \leq s' \stackrel{\text{def}}{\Longleftrightarrow} s \sqsubseteq s' \end{cases}$$

- Generate について

  文献 [6]の (Limit) を適用する。

$$\begin{cases} C_G \stackrel{\text{def}}{\Longleftrightarrow} \text{ある } u \in U \text{ について } u \xrightarrow{\gamma} s \text{ というパスが存在し、} s = \bar{\gamma}(u) \text{ かつ } u \leq t \\ t_G \stackrel{\text{def}}{=} rep(\bigvee_{i \in \mathbb{N}} \bar{\beta}^i(u)) \quad (\beta \stackrel{\text{def}}{=} \gamma; a) \end{cases}$$

- U と S からの要素の削除

  文献 [6]と同じにするならば Generate, Step において U と S から要素は取り除かない。ただし、以下で述べる性質は U と S からの要素の削除如何には

関わらない。

文献 [6]では具体状態上のパスでは性質 1 の条件が成り立つようにシステムの前提条件が定められている。このとき、以下の性質が成り立つ。

系 初期状態から始まる具体状態上の任意の経路

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n$$

について、アルゴリズムが終了した時点で $s_n \leq s'$ となる状態 $s' \in U$ が存在する。

これは、具体的なシステムにおいてもし危険な状態に到達する経路が存在すれば、抽象到達性検査でその状態よりも一般的な状態に必ず到達するということを示している。

7.2 抽象グラフ探索アルゴリズム

3.4節で定義し、正当性を検証した抽象グラフ探索アルゴリズム (図 3) も、実は抽象到達性検査アルゴリズムで表現することができる。

- ラベル付き遷移システム

$$\begin{cases} \mathcal{S} \stackrel{\text{def}}{=} V \times L \\ \mathcal{A} \stackrel{\text{def}}{=} E \\ s \xrightarrow{a} t \stackrel{\text{def}}{\Longleftrightarrow} \exists v\, w\, x.\ s = (v,x),\ t = (w, f_e(x)),\ v \xrightarrow{e} w,\ \text{and } a = e \\ \mathcal{I} \stackrel{\text{def}}{=} \{(v_s, b_s)\} \\ s \leq s' \stackrel{\text{def}}{\Longleftrightarrow} \exists v\, x\, x'.\ s = (v,x),\ s' = (v,x'),\ \text{and } x \geq x' \end{cases}$$

- Generate について ($s = (v,x)$, $t = (w, f_e(x))$ とする)

$$\begin{cases} C_G \stackrel{\text{def}}{\Longleftrightarrow} \bigwedge\{y \mid (w,y) \in U\} \not\geq f_e(x) \\ t_G \stackrel{\text{def}}{=} (w, f_e(x) \wedge \bigwedge\{y \mid (w,y) \in U\}) \end{cases}$$

- U と S からの要素の削除

  Generate, Step において、U と S から取り除けるものは全て取り除く。

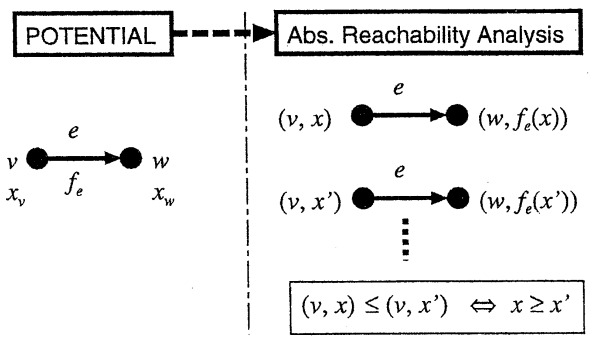

図 7 抽象グラフ探索と抽象到達性検査との関係

状態間の順序 ($s \leq s'$) と抽象グラフ探索で頂点に割り当てられている値の間の順序 ($x \geq x'$) が逆転していることに注意。ここでは小さい値を割り当てられている頂点の方がより抽象的と考える。

実は、Generate, Step において、U と S から取り除けるものは全て取り除くと、v に対して $s = (v,x) \in U$ となる s は高々一つである。よって、上の $\bigwedge\{y \mid (w,y) \in U\}$ の部分は実質的には「**if** $\exists (w,y) \in U$ **then** y **else** $\top$」になる。

$s \xrightarrow{a} t$ かつ $s \leq s'$ のとき、すなわち、$s = (v,x)$, $s' = (v,x')$, $t = (w, f_a(x))$ で $x \geq x'$ のとき、$t' = (w, f_a(x'))$ とおくと、f_a の単調性から $s' \xrightarrow{a} t'$ かつ $t \leq t'$ が成り立つ。つまり、任意の有限長の経路について、性質 1 の条件部分は満たされる。よって、抽象グラフ探索では、性質 1 の結論部分、すなわち以下の性質が成り立つことが言える。

系 G 中の、開始点から始まる任意の経路

$$v_s \equiv v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \ldots \xrightarrow{e_{n-1}} v_n$$

について、アルゴリズム終了時に、$(v_n, x) \in U$ となる x が存在し、

$$x \leq (f_{e_{n-1}} \circ \ldots \circ f_{e_1} \circ f_{e_0})(b_s)$$

が成り立つ。

これは、アルゴリズム終了時に求まった値が、経路解よりも小さいということを表している。なお、この証明では f_e の単調性しか用いておらず、3.4節の証明ではアルゴリズム終了時に求まった値が経路解と等しいことを示すのにさらに f_e が分配的であることを仮定しなければならなかったことに注意。

8 おわりに

本論文に現れた様々なアルゴリズムの間の関係を模式的に表したのが図 8である。

まず、グラフの到達可能性問題や最短路問題などを包括する抽象グラフ探索アルゴリズム POTENTIAL の形式的検証を証明検証系 HOL を用いて行った。この検証結果は最短路問題の最適化アルゴリズムの一つである A* アルゴリズムの検証に利用され、抽象的アルゴリズムの検証から具体的なアルゴリズムの検証を得ることに成功した。

また、グラフの到達可能性を別の視点から抽象化した抽象到達可能性アルゴリズムを定義し、モデル検査のアルゴリズムを抽象化した Covering Graph 作成や、抽象グラフ探索アルゴリズムがその例に

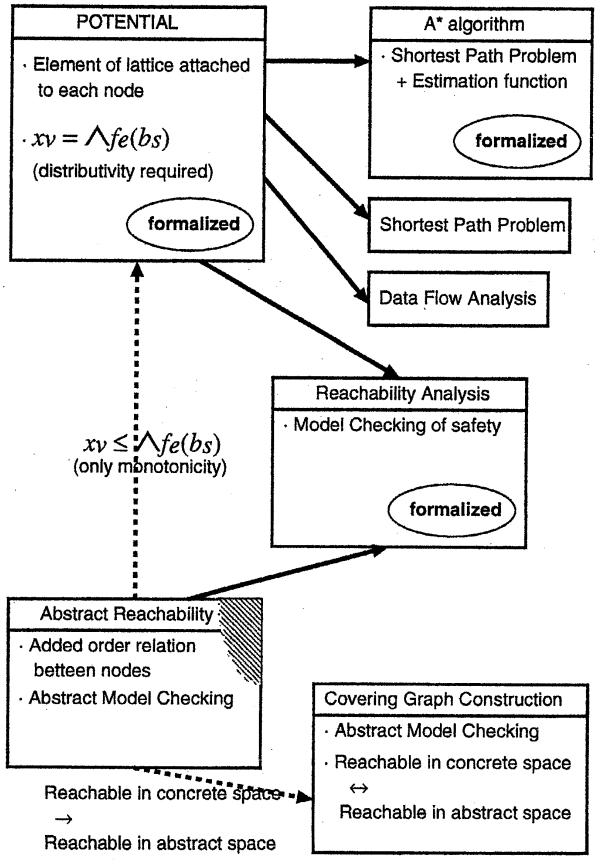

図 8　全体の構造

なっていることを示した。ただし、これらの例について今のところ示した性質は図 8にあるようにいずれも一部分のみであり、抽象到達可能性はまだ完全であるとは言えない (その意味で図の中では影が付いている)。証明検証系による形式的検証と合わせて、この部分は将来の課題である。

参考文献

[1] Mark D. Aagaard, Robert B. Jones, and Carl-Johan H. Seger. Lifted-FL: A pragmatic implementation of combined model checking and theorem proving. In *Theorem Proving in Higher Order Logics: TPHOLs '99*, volume 1690 of *LNCS*, pages 323–340. Springer-Verlag, 1999.

[2] Karthikeyan Bhargavan, Carl A. Gunter, Elsa L. Gunter, Michael Jackson, Davor Obradovic, and Pamela Zave. The village telephone system: A case studyj in formal software engineering. In *Theorem Proving in Higher Order Logics: TPHOLs '98*, volume 1479 of *LNCS*, pages 49–66. Springer-Verlag, 1998.

[3] Albert J Camilleri and Wishnu Prasetya. Cpo theory, 1994. Available from `http://www.cl.cam.ac.uk/ftp/hvg/hol88/contrib/cpo/`.

[4] Ching-Tsun Chou and Doron Peled. Verifying a model-checking algorithm. In *Tools and Algorithms for the Construction and Analysis of Systems*, number 1055 in LNCS, pages 241–257, Passau, Germany, 1996. Springer-Verlag.

[5] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[6] E. Allen Emerson and Kedar S. Namjoshi. On model checking for non-deterministic inifinite-state systems. In *13th Annual IEEE Symposium on Logic in Compuer Science*, pages 70–80, 1998.

[7] M. J. C. Gordon and T. F. Melham, editors. *Introduction to HOL: A theorem proving environment for higher order logic*. Cambridge University Press, 1993.

[8] G. J. Holzmann. *Design and Validation of Computer Protocols*. Prentice Hall, 1991.

[9] G. J. Holzmann and D. Peled. An improvement in formal verification. In *7th International Conference on Formal Description Techniques*, pages 177–194, 1994.

[10] G. Kildall. A unified approach to global program optimization. In *POPL*, pages 194–206, 1973.

[11] Bruno Martin, July 1997. Private Communication.

[12] Nils J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing, 1980.

[13] N. Shankar, S. Owre, and J. Rushby. The PVS proof checker: A reference manual. Technical report, Computer Science Lab, SRI Intl., 1993.

[14] Tetsuo Tamai. A class of fixed-point problems on graphs and iterative solution algorithms. In *Logic and Software Engineering*, pages 102–121. World Scientific, 1996.

[15] W. Wong. A simple graph theory and its application in railway signalling. pages 395–409.