古典的プランニングにおける Axiom 自動抽出について

On Automatic Axiom-Extraction in Classical Planning

三浦脩和 [1] *　堀江慧 [2]　福永 Alex[2]
Shuwa Miura　Satoru Horie　Alex Fukunaga

[1] 東京大学教養学部
College of Arts and Sciences, The University of Tokyo
[2] 東京大学大学院総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo

Abstract: Axioms can be used to model derived predicates in domain-independent planning models. Formulating models which use axioms can sometimes result in problems with much smaller search spaces than the original model. We propose a method for automatically extracting a particular class of axioms from standard STRIPS PDDL models. More specifically, we identify operators whose effects become irrelevant given some other operator, and generate axioms that capture this relationship. We show that this algorithm can be used to successfully extract axioms from standard IPC benchmark instances, and show that the extracted axioms can be used to significantly improve the performance of an IP-based planner.

1　はじめに

人工知能分野におけるプランニングとは、エージェントの行動の自動決定のためのモデルベースの手法である [5]。エージェントの行動 (オペレータ) がいつ実行可能であるか (前提条件)、状態をどのように遷移させるか（効果）を定めたモデルが与えられたとき、初期状態から、与えられた条件を満たす終了状態に遷移させるオペレータの列（プラン）を求める。古典的プランニング問題は STRIPS[3]、SAS+[1] といった言語で記述される。STRIPS、SAS+で表されたプランニング問題の解を探す問題は PSPACE 完全である [2]、[1]。図 1a に示した Miconic ドメインにおいて、乗客が上のフロアにいる状態が初期状態であり、乗客が下のフロアに到着した状態が終了状態である。エレベータの乗り降り、エレベータの上下動といったオペレータを組み合わせて、初期状態から終了状態に到達することを目指す。

STRIPS、SAS+は状態の特徴をオペレータの適用の結果として定める。しかし、ある種の特徴はオペレータの適用の直接の結果ではなく、その他の特徴から導かれる副次的な特徴としてより自然に表される。こうした副次的な特徴を表すため、STRIPS、SAS+は axiom と呼ばれる状態の特徴を 表すルールを含むことができる。[9] は一部のドメインにおいて、axiom を含む方が含まない場合に比べて、表されるプランニング問題の状態空間が小さくなることを示した。例えば、Miconic ドメインに「乗客をのせたエレベータが上のフロアに到達可能ならば、下のフロアにも到達可能である。」という axiom を含めると、エレベータの到達可能性を他の特徴から導くことができる。このとき、エレベータの上下動等をオペレータではなく、状態内でエレベータの到達可能性を表す axiom して表すことで、冗長な状態の遷移がなくなり、状態空間が図 1b のように小さくなる。

しかし、あるプランニング問題が axiom を用いてよりよく表されるかは必ずしも明らかではない。プランニングがエージェントの行動の自動決定であることを考えると、問題の最適な記述を要求するのではなく、よりよい記述に自動変換できることが望ましい。本論文では、SAS+で表された問題を axiom を含む記述に自動変換する方法を新たに提案する。変換後の問題を IP 問題への変換で解いた結果、変換前の問題に比べて解けた問題数が増加した。

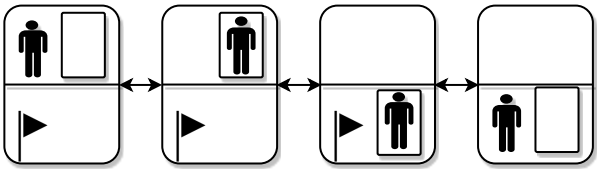
2　SAS+と Axiom

SAS+は多値変数への値の割り当てによってプランニング問題の状態を記述する。

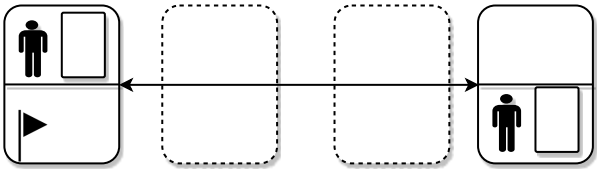
定義. SAS+問題 Π は (V, O, I, G, U, A) の組からなる。

- V=$\{v_1, ..., v_n\}$ は主変数の集合。主変数はそれぞれ定義域 $D(v_i)$ を持つ。

*連絡先：9490751716@mail.ecc.u-tokyo.ac.jp



(a) axiom を用いない場合の状態空間



(b) axiom を用いた場合の状態空間

図 1: Miconic 問題の状態空間。人型は乗客の 現在地、旗は乗客の目的地を表す。箱はエレベータに相当する。

- O はオペレータの集合。o∈O は、前提条件 pre(o)、効果 eff(o)、コスト cost(o) からなる。前提条件と効果は、変数上の部分割り当てであり、$\{v_i$=x, v_j=$y\}$ の形をとる。オペレータ o は pre(o) を満たす状態 s で適用可能である。状態 s に eff(o) を割り当てた状態を s'、s' から副次変数の割り当てを除いた部分割り当てを p、p に対して A に含まれる axiom を評価した状態を A(p) とする。状態 s にオペレータ o を適用して得られる状態は A(p) である。
- I は主変数の完全割り当てであり、初期状態は $A(I)$ になる。
- G は変数の部分割り当てであり、終了状態。
- U=$\{u_1, ..., u_m\}$ は副次変数の集合。副次変数はいかなるオペレータの効果に現れない。変数はそれぞれ定義域 $D(u_i)$=$\{0, 1\}$ を持つ。
- A は axiom の集合。a∈A はヘッド head(a)、ボディ body(a) からなる。ヘッドとボディは、変数上の部分割り当て $\{v_i$=x, v_j=$y, ...\}$ である。ボディは一つの副次変数 u_i=1 の割り当てのみからなる。axiom は全ての状態において、ヘッドが成り立てば、ボディも成り立つことを要求する。ただし、副次変数の値は axiom によって 1 にされない限りデフォルトでは 0 である。axiom は u_i=1←v_j=x, v_l=y の形をとる。

Axiom 集合 A は階層化 (stratified) されていなければならない。A は以下の条件を満たす U から $\{0, ..., m\}$ への写像 l が存在するとき階層化されている。

- u_i=1 をヘッドに持つ axiom a∈A について、a のボディに現れる u_j=1 について、$l(u_j)$≤$l(u_i)$。
- u_i=1 をヘッドに持つ axiom a∈A について、a のボディに現れる u_j=0 について、$l(u_j)$<$l(u_i)$。

A が階層化されているとき、A(p) は必ず一意に定まる。

SAS+問題の解、またはプランとは状態列 $s_0, ..., s_{n+1}$ (s_{n+1} は終了状態) を生み出すオペレータの列 $o_0, ..., o_n$ である。プランのコストはプランに含まれるオペレータのコストの和で定義される。プランが最適であるとは、プランのコストが最小であることである。

実際のプランニング問題は変数の利用等、より高機能な PDDL[6] で記述されていることが多い。FastDownward[7] 等、一部のソルバは [8] の方法を用いて、PDDL 問題を SAS+問題に変換してから探索を行う。axiom の存在する PDDL 問題に対する手法として、axiom をオペレータに変換して解く手法がある。しかし、[13] はこの変換がプランのサイズ、問題のサイズを多項式に抑えられないことを示した。Fastdownward 等の経路探索に基づくソルバは、多くの場合、各状態ごとに axiom の評価を行う。図 1a の Miconic 問題に対応する axiom を含まない SAS+問題 Π=$(V, O, I, G, U$=ϕ, A=$\phi)$ は次のようになる。

- V=$\{v_0, v_1, v_2\}$ は乗客が搭乗しているか ($D(v_0)$=$\{\neg$boarded, boarded$\}$)、エレベータの場所 ($D(v_1)$=$\{$at(f0), at(f1)$\}$)、乗客が目的地に到着したか ($D(v_2)$=$\{\neg$served, served$\}$) を表す変数からなる。
- O はエレベータの乗り降り (board、depart) とエレベータの上下動 (up、down) を表す以下のオペレータからなる。pre(board) = $\{v_1$=at(f1)$\}$、eff(board)= $\{v_0$=boarded$\}$、pre(depart)=$\{v_0$=boarded,v_1=at(f0)$\}$、eff(depart) =$\{v_0$=$\neg$boarded ,v_2=served$\}$、pre(down)=$\{v_1$=at(f1)$\}$、eff(down) =(v_1=at(f0)$\}$、pre(up)=$\{v_1$allowbreak=at(f0))、eff(up)=(v_1=at(f1))$\}$。
- I=$\{v_0$=$\neg$boarded,v_1=at(f1)$\}$、G=$\{v_2$=served$\}$。

up、down、board を axiom にした SAS+問題 Π′=$(V'$, $O', I', G', U', A')$ は次のようになり、図 1b に対応する。

- U' は定義域 {0,1} となる u_3, u_4, u_5, u_6 からなる。
- u_3, u_4, u_5, u_6 は主変数の到達可能性を表す。
  - u_3=1←u_0=boarded, u_1=at(f0)
  - u_4=1←u_0=boarded, u_1=at(f1)
  - u_5=1←u_0=$\neg$boarded, u_1=at(f0)
  - u_6=1←u_0=$\neg$boarded, u_1=at(f1)
- O' は depart のみからなる。pre(depart)=(u_3=1)、eff(depart)=(v_0=$\neg$boarded,v_1=at(f0),v_2=served)。
- A' は加えて次の axiom を含む。u_3=1←u_4=1、u_4=1←u_3=1、u_4=1←u_6=1、u_5=1←u_6=1、u_6=1←u_5=1。

3 Axiom の自動抽出

axiom を含まない SAS+問題 Π を axiom を含む SAS+問題 Π′ に変換することを SAS+Axiom 変換と呼ぶ。変

換された問題 Π' は元の問題 Π が解を持つとき、またそのときに限り解を持つ。本節では、オペレータの一部を axiom に変換することで SAS+Axiom 変換する手法を新たに提案する。

Miconic ドメインの問題では、up、down、board オペレータを axiom として表すことができた。これは、up、down、board の効果に含まれる変数 var0 が depart の前提条件に含まれるためである。このとき、up、down、board をどのように繰り返しても、depart オペレータを適用したときに、up、down、board で変更し得る var0 の値が決定する。一般に、問題 Π のオペレータ集合 O を $V_{\mathrm{eff}}(O_{\mathrm{sub}}) \subset V_{\mathrm{pre}}(O_{\mathrm{dom}})$ となる subsumed operators(O_{sub}) と dominant operators(O_{dom}) に分割できるときに、O_{sub} を axiom に変換できる。ただし、$o \in O$ について、効果に現れる変数の集合を eff_var(o)、前提条件に現れる変数の集合を pre_var(o) で表し、$V_{\mathrm{eff}} = \bigcup_{o \in O}$ eff_var(o)、$V_{\mathrm{pre}} = \bigcap_{o \in O}$ pre_var(o) である。提案する SAS+Axiom 変換の手法は、まずオペレータを頂点とするグラフを用いることで、オペレータが O_{sub} と O_{dom} に分割できるか調べる。次に、$V_{\mathrm{eff}}(O_{\mathrm{sub}})$ に含まれる変数の部分割り当てが到達可能かを表す副次変数を導入することで、axiom を含む問題に変換する。

3.1 オペレータグラフ

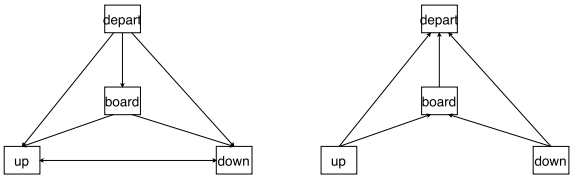
問題 Π のオペレータ集合 O を O_{sub} が包含関係において極大となる分割 O_{sub}、O_{dom} に分割することを考える。このような分割 O_{sub}、O_{dom} を効率的に求めるために、オペレータグラフを定義する。

定義. SAS+問題 $\Pi=(V, O, I, G)$ に対するオペレーターグラフ $G=(V, E)$ は次の性質を有する有効グラフである：(1) $V=O$、(2) eff_var(o_1) $\subset$ pre_var(o_2) のとき、またそのときに限り $(o_1, o_2) \in E$。

オペレータグラフ G が、O_2 の全ての頂点から O_1 に含まれる全ての頂点に辺が存在するように分割できるとき、分割 O_1、O_2 は $V_{\mathrm{eff}}(O_1) \subset V_{\mathrm{pre}}(O_2)$ を満たす。このとき、O_1、O_2 はそれぞれ O_{sub}、O_{dom} に対応する。Miconic 問題インスタンスに関して構築したオペレータグラフを、 図 2a に示した。このとき、オペレータ集合は O_1={up,down,board}、O_2={depart} に分割でき、それぞれ O_{sub}、O_{dom} に対応する。

3.2 オペレータグラフの分割

与えられた有向グラフ G の頂点集合が一方の全ての頂点から、もう一方の全ての頂点に辺が存在する頂点集合に分割できるかどうかは、補グラフ $\overline{G}$ の連結性を調べることで判定できる。これは、無向グラフの頂点



(a) オペレータグラフ　　(b) 補グラフ

図 2: Miconic 問題のオペレータグラフとその補グラフ

集合が完全二部グラフをなす二つの頂点集合に分割可能か判定するアルゴリズム [4] と同様である。

有向グラフ G が与えられたとき、O_2 の全ての頂点から O_1 の全ての頂点に辺が存在する分割 O_1、O_2 のうち、O_1 が包含関係において極大となるものを求めるには、まず G の補グラフ $\overline{G}$ を構築して $\overline{G}$ の強連結成分を求める (Tarjan のアルゴリズム [12] など $\mathcal{O}(|V|+|E|)$ のアルゴリズムが存在する)。強連結成分のグラフで、出次数が 0 の強連結成分に含まれる頂点集合を O_2、それ以外の頂点集合を O_1 とすると、O_2 の全ての頂点から O_1 の全ての頂点に辺が存在する。これは、補グラフ上で O_2 に含まれる頂点から O_1 に含まれる頂点への辺がないためである。

グラフ上の辺の数は高々$|V|^2$ 個であるため、分割 O_1、O_2 は $\mathcal{O}(|V|^2)$ の実行時間で求めることができる。こうして求めた O_1 は包含関係において極大である。なぜなら、$O_1 \subsetneq O_1'$ となる頂点集合を考えると、$o \in O_1' \wedge o \in O_2$ となる頂点が存在する。このとき補グラフにおいて、O_2 が強連結成分であることから、ある $o' \in O_2$ が存在し辺 (o', o) が存在する。このとき、元のグラフにおいて辺 (o', o) が存在しないため、O_2 に含まれる全ての頂点から全ての頂点への辺が存在するという性質を満たさない。Miconic 問題のオペレータグラフでは、補グラフの強連結成分はそれぞれ元のグラフの頂点であり、O_2={depart} となる。

3.3 エンコード

SAS+問題 $\Pi=(V, O, I, G, U=\phi, A=\phi)$ を受け取り、axiom を含む SAS+問題 $\Pi'=(V', O', I', G', U', A')$ を以下の**エンコード**と呼ぶ手続きにより構成する。ただし、O は O_{sub} と O_{dom} に分割できるものとする。

- $V'=V$、$I'=I$。
- $v_1, ..., v_n \in V_{\mathrm{eff}}(O_{\mathrm{sub}})$、$x_1 \in D(v_1), ..., x_n \in D(v_n)$ それぞれについて $a_{\{v_1=x_1, ..., v_n=x_n\}} \in U'$。また、$a_{\{v_1=x_1, ..., v_n=x_n\}}=1 \leftarrow v_1=x_1, ..., v_n=x_n \in A'$。
- $v_i=x \in G$ について、$v_i \notin V_{\mathrm{eff}}(O_{\mathrm{sub}})$ ならば、$v_i=x \in G'$。$v_i \in V_{\mathrm{eff}}(O_{\mathrm{sub}})$ となる変数を集めた $\{v_1=$

$x'_1, ..., v_m{=}x'_m\}$ について、$a_{\{v_1=x'_1,...,v_m=x'_m\}} \in U'$、$a_{\{v_1=x'_1,...,v_m=x'_m\}}{=}1 \in G'$。$\{v_1{=}x'_1, ..., v_m{=}x'_m\} \subset \{v_1{=}x_1, ..., v_n{=}x_n\}$ について、$a_{\{v_1=x'_1,...,v_m=x'_m\}} {=}1 \leftarrow a_{\{v_1=x_1,...,v_n=x_n\}}{=}1 \in A'$。

- $o \in O_{\mathrm{dom}}$ について、以下のような $o' \in O'$。
  - $v_i{=}x \in \mathrm{pre}(o)$ について、$v_i \notin V_{\mathrm{eff}}(O_{\mathrm{sub}})$ ならば、$v_i{=}x \in \mathrm{pre}(o')$。$V_{\mathrm{eff}}(O_{\mathrm{sub}}) \subset \mathrm{pre\_var}(o)$ より、$v_i \in V_{\mathrm{eff}}(O_{\mathrm{sub}})$ となる変数を集めた $\{v_1{=}x_1, ..., v_n{=}x_n\}$ について、$a_{\{v_1=x_1,...,v_n=x_n\}} {=}1 \in \mathrm{pre}(o')$。
  - $v_i{=}x \in \mathrm{eff}(o)$ について、$v_i{=}x \in \mathrm{eff}(o')$。$v_i{=}x \in \mathrm{pre}(o)$ について、$v_i \in V_{\mathrm{eff}}(O_{\mathrm{sub}})$ かつ、v_i に関する割り当てが $\mathrm{eff}(o)$ に含まれないならば、$v_i{=}x \in \mathrm{eff}(o')$。
- $v_1, v_2, ..., v_n \in V_{\mathrm{eff}}(O_{\mathrm{sub}})$ について、$o \in O_{\mathrm{sub}}$ について、o の前提条件が $v_1, v_2, ..., v_n$ に関して、$\{v_1{=}x_1, ..., v_n{=}x_n\}$ で成り立つなら、$a_{\{v_1=x'_1,...,v_n=x'_n\}} {=}1 \leftarrow a_{\{v_1=x_1,...,v_n=x_n\}}{=}1, v_{n+1}{=}w, ... \in A'$。ただし、$v_{n+1}, ...$ は $v_1, ..., v_n$ 以外の変数に関する前提条件であり、$\{v_1{=}x'_1, ..., v_n{=}x'_n\}$ は o を適用したときに得られる、$v_1, ..., v_n$ に関する部分割り当てである。

エンコードは以下の性質を持つ。

定理. 状態 s について $A(s)$ において $a_{\{v_1=x_1,...,v_n=x_n\}}{=}1$ となるとき、またそのときに限り、変換前の問題で、s から O_{sub} に含まれるオペレータのみを使って、$\{v_1{=}x_1, ..., v_n{=}x_n\}$ の部分割り当てに到達できる。

証明. $a_{\{v_1=x_1,...,v_n=x_n\}=1}$ のとき、変数を真にする axiom の列が存在する。この axiom に対応するオペレータを適用することで、変換前の問題で、s から $o \in O_{\mathrm{sub}}$ のみを使って、$\{v_1{=}x_1, ..., v_n{=}x_n\}$ に到達できる。

逆に、変換前の問題で、s から $o \in O_{\mathrm{sub}}$ のみを使って、$\{v_1{=}x_1, ..., v_n{=}x_n\}$ に到達できるとき、オペレータに対応する axiom の列が存在し、$a_{\{v_1=x_1,...,v_n=x_n\}}{=}1$ となる。

定理. 変換された問題 Π' は元の問題 Π が解を持つとき、またそのときに限り解を持つ。

証明. 元の問題 Π が解 $p{=}o_1, ..., o_n$ を持つとき、p から $o \in O_{\mathrm{sub}}$ を取り除いたオペレータに対応する $p'{=}o'_1, ..., o'_m$ は変換された問題 Π' の解である。Π' のオペレータの構成の仕方から、対応する o_i、o'_j について、$\mathrm{s}{=}o_i(...(o_1(I))) {=}o'_j(...(A(o'_1(I))))$。$o_i$ が適用可能であることから、s で o'_j の前提条件は $a_{\{v_1=x_1,...,v_n=x_n\}=1}$ 以外は成り立っている。また元のプランで s から $o \in O_{\mathrm{sub}}$ のみを使って他の変数の割り当てを変更せずに $\{v_1{=}x_1, ..., v_n{=}x_n\}$ に到達できることから、上に示した通り、$a_{\{v_1=x_1,...,v_n=x_n\}=1}$ となり、o'_j が適用できる。同様に解が終了状態に到達することを示すことができる。$p'{=}o'_1, ..., o'_m$ は変換された問題 Π' の解である。

変換後の問題 Π' が解 $p{=}o'_1, ..., o'_m$ を持つとき、対応するオペレータの列 $o_1, ..., o_m$ が得られる。初期状態にこのオペレータの列に含まれるオペレータを適用していく。オペレータの前提条件、または問題の終了状態に含まれる条件 $\{v_1{=}x_1, ..., v_n{=}x_n\}$ が成り立たないたび、現在の状態を初期状態、$\{v_1{=}x_1, ..., v_n{=}x_n\}$ を終了状態とする SAS+問題の解を挿入する。この部分問題のオペレータは O_{sub} 限定される。変換後の問題で $a_{\{v_1=x_1,...,v_n=x_n\}=1}$ が成り立つことから、この部分問題は必ず解をもつ。こうして構成された $p{=}o_1, ..., o_n$ は元の問題 Π の解である。

3.4 デコード

先に述べたように、変換後の問題 Π' の解 $p'{=}o'_1, ..., o'_m$ から、条件 $\{v_1{=}x_1, ..., v_n{=}x_n\}$ が成り立たないたび、現在の状態を初期状態、$\{v_1{=}x_1, ..., v_n{=}x_n\}$ を終了状態とする部分問題の解を挿入することで、元の問題 Π の解 p が得られる。この手続きを**デコード**と呼ぶ。

SAS+Axiom 問題における axiom は重みを持たない。そのため、分割 O_{sub}、O_{dom} に関して、SAS+問題 Π を SAS+Axiom 問題 Π' にエンコードする際、$o \in O_{\mathrm{sub}}$ のコストに関する情報は失われる。このとき、Π' の最適解を最適探索を用いてデコードしても、元の問題 Π の最適解が得られるとは限らない。ただし、O_{sub}、O_{dom} において、$\forall_{o \in O_{\mathrm{sub}}} cost(o){=}0$ ならば、デコード結果は元の問題 Π の最適解となる。

デコードは $\mathrm{s}{=}o_1, ..., o_k(I)$ を初期状態、$\{v_1{=}x_1, ..., v_n{=}x_n\}$ を終了状態とする部分問題を繰り返し解く。このとき、それぞれの部分問題は SAS+問題であることから、PSPACE 完全である。ただし、現実的には $V_{\mathrm{eff}}(O_{\mathrm{sub}}$ に含まれる変数が限定されることから部分問題は極めて簡単であった。

4 SAS+Axiom 変換の評価

本節では、提案手法である SAS+Axiom 変換を International Planning Competition(IPC) の IPC1998 から IPC2011 で扱われた問題に適用して評価する。全ての実験は 2.3GHz の Xeon CPU を用いて行った。実験で用いるプランニング問題は全て PDDL で記述されている。SAS+問題を入力とするプログラムには、PDDL を FastDownward で SAS+問題に変換し、入力を与えた。

ただし、FastDownward から出力される SAS+問題が元々axiom を含む問題は除いている。実行時間 5 分、メモリ 2GB の制限で行った。エンコーダは Python2.7.6 で書かれ、CPython で実行された。ただし、計算時間、

メモリ使用量のボトルネックとなるオペレータグラフの構築と完全二部グラフへの分解は C++のグラフアルゴリズムライブラリ lemon-1.3.1 を利用した。全 1672 問中、1220 問が制限内に終了し、そのうち Miconic ドメインや gripper ドメインの問題等 249 問で axiom が抽出された。実行時間、メモリ使用量は概ね問題インスタンスのオペレータ数に依存した。gripper 等オペレータ数の少ないドメインでは、極めて短い時間で SAS+Axiom 変換が終了した。しかし、parking ドメイン等、オペレータ数の多いドメインでは、メモリ制限を超過した。

一方、Sokoban 等、直感的には axiom へ変換できるはずのドメインで axiom が抽出されていない。Sokoban はプレイヤーが石をそれぞれのゴールまで押す手順を求めるパズル問題である。clear(?loc) はある場所 loc が空いていることを意味するが、直感的にはある場所が空であるかは、石とプレイヤーの場所から一意に決まるため、clear(?loc) は冗長な変数といえる。この冗長な変数が原因で、O_{sub} と O_{dom} への分割ができず、axiom の抽出ができない。

5 Axiom 対応化した IP プランナでの評価

本節では SAS+Axiom 変換後の問題を axiom に対応した IP プランナで解いて、性能を評価する。STRIPS、SAS+問題は SAT、IP、CSP 問題への長さ n 変換によって解くことができる。問題 Π と自然数 n を受け取り、問題 Π′ を返す関数は問題を長さ n 制限で変換するという。変換された問題 Π′ は元の問題 Π が長さ n 以下の解 (プラン) を持つとき、またそのときに限り解を持つ。最初にある長さ n で解を求めて、解が見つかるまで n を増やして、この手続きを、解が見つかるまで、あるいは資源制限 (時間・メモリ) を越えるまで繰り返す。SAT、IP、CSP への変換によってプランニング問題を解くプランナをそれぞれ SAT、IP、CSP プランナと呼ぶ。

State Change Variable モデル [14] は STRIPS 問題を、命題の真偽値ではなく、命題の真偽値の変化を表す IP 問題に長さ n 制限変換する。SAS+問題は変数の割り当て $v_i{=}x$ を命題 $f(v_i{=}x)$ に変換することで、自動的に STRIPS 問題に変換することができることから、同様に IP 問題に長さ n 制限変換できる。[16] で開発されたプランナ (hp プランナ) は State Change Variable モデルの上に Optiplan[10] で提案された planning graph の簡易版 relaxed planning graph の利用し、新たに SAS+の多値変数の値の mutex 制約を導入した。

既存の SAT、IP、CSP プランナは axiom に対応していない。紙面の都合により、IP プランナを axiom に対応させる方法の概要のみ簡単に説明する。詳細は [15] を参照のこと。本手法は [9] が提案したように、SAS+問題の axiom の評価を解集合プログラミング (ASP) という宣言的な論理プログラミングの問題として解く。[11] が提案した ASP から IP への変換と組み合わせることで、axiom を含む SAS+問題を IP へ長さ n 制限変換をする。変換後の問題は主変数に関して、State Change Variable モデルと同じ変数、制約を持つ。また、ステップ t で対応する命題が真であることを意味する補助的な変数 $x^{\mathrm{sat}}_{f,t}$ を導入する。$a{\leftarrow}b_1,...,b_m,\mathrm{not}c_1,...,\mathrm{not}c_m$ の形の axiom について、ステップ t における axiom からなる ASP プログラム P_t を考える。P_t は以下のルールからなる。

$$x^{\mathrm{sat}}_{a,t}{\leftarrow}x^{\mathrm{sat}}_{b_1,t},...,x^{\mathrm{sat}}_{b_n,t},\mathrm{not}x^{\mathrm{sat}}_{c_1,t},...,\mathrm{not}x^{\mathrm{sat}}_{c_m,t} \qquad (1)$$

P_t の解はそのままステップ t における副次変数の真偽値に対応する。各ステップの P_t を IP に変換する。さらに、変換後の問題は、各ステップにおいて実行されるオペレータはただ一つであるという制約を含む。axiom が含まれない State Change Variable モデルでは、互いに矛盾しないオペレータは同じステップで並列実行することが可能であった。しかし、axiom が含まれる問題では、主変数の変化に矛盾を引き起こさないオペレータであっても、副次変数に矛盾を生じることがある。

hp プランナで SAS+問題から変換された IP 問題は Gurobi Optimizer 6.5.0 で解いた。IP 問題には目的関数を与えず、 制約を満たす整数ベクトルを求めた。総実行時間 30 分（１スレッド）、メモリ 2GB の制限の下で実行した。SAS+Axiom 変換は問題毎に行われることから、総実行時間には、PDDL を SAS+問題に変換、エンコード、SAS+Axiom 問題の解を探索、デコードする時間の全てが含まれる。変換後の問題の解をデコードするには、元の問題の部分問題を解く必要があるが、この部分問題は FastDownward の A*探索で解いた。部分問題は、現実的には簡単な問題であるため、終了状態には 0、その他の状態にはコストの最も低い

	mutex+pg		mutex+axiom	
ドメイン (問題数)	Coverage	n	Coverage	n
airport(6)	6	18	6	16
grid(2)	1	14	1	4
gripper(20)	5	15	12	16
miconic(150)	34	10	76	7
pegsol-sat11(20)	3	20	19	14
scanalyzer-08(3)	2	5	2	4
tpp(4)	4	5	4	7
visitall-sat11(14)	0	None	0	None

表 1: IP プランナにおける SAS+Axiom 変換前後の実行結果比較。Coverage は解けた問題数の総和。n は全ての設定で解けた問題において、解が見つかった n 制限の平均 (丸め)。None は該当するデータが存在しないことを表す。

オペレータのコストを返す blind ヒューリスティックを用いた。SAS+Axiom 変換が 適用可能であったドメインにおいて SAS+Axiom 変換なし (mutex+pg)、SAS+Axiom 変換あり (mutex+axiom) 設定での実験結果を表 1 にまとめた。ただし、mutex+axiom では relaxed planning graph を単純に適用できないため使用していない。mutex+axiom で、gripper、Miconic、pegsol ドメインにおいて解けた問題数が著しく増加した。これらのドメインでは多くのオペレータが axiom に変換され、解が見つかった長さ n が平均して小さくなっている。ただし、gripper ドメインでは多くのオペレータが axiom に変換されたものの、オペレータの並列実行ができなくなったために n がわずかに増加している。

6 おわりに

本論文では、SAS+問題を axiom を含む問題にに自動変換する新規手法を提案した。実際に Miconic 等で自動変換ができることを示した。しかし、Sokoban 等、冗長な変数が原因で提案手法では変換できないドメインも存在する。こうしたドメインを自動変換することは今後の課題である。

さらに、変換前後の問題を axiom に対応するプランナに解かせることで、SAS+Axiom 変換の有用性を評価した。IP ベースプランナの実験では、変換後の問題を解くことで解けた問題数が増加した。しかし、axiom に変換されるオペレータが少ない問題では、解けた問題数は増えなかった。どのような問題に対して、SAS+Axiom 変換を適用すべきか判断するのは今後の課題である。また、IP ではなく SAT や ASP を用いたプランナを axiom に対応させて解くことも考えられる。

参考文献

[1] C. Bäckström and B. Nebel. Complexity results for SAS+ planning. *Computational Intelligence*, 11(4):625–655, 1995.

[2] T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1):165–204, 1994.

[3] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3):189–208, 1972.

[4] H. Fleischner, E. Mujuni, D. Paulusma, and S. Szeider. Covering graphs with few complete bipartite subgraphs. *Theoretical Computer Science*, 410(21):2045–2053, 2009.

[5] H. Geffner and B. Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers, 2013.

[6] M. Ghallab, C. Knoblock, D. Wilkins, A. Barrett, D. Christianson, M. Friedman, C. Kwok, K. Golden, S. Penberthy, D. E. Smith, et al. PDDL-the planning domain definition language. 1998.

[7] M. Helmert. The Fast Downward planning system. *J. Artif. Intell. Research*, 26:191–246, 2006.

[8] M. Helmert. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173(5):503–535, 2009.

[9] F. Ivankovic and P. Haslum. Optimal planning with axioms. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1580–1586. AAAI Press, 2015.

[10] S. Kambhampati and M. H. L. van den Briel. Optiplan: Unifying IP-based and graph-based planning. *J. Artific. Intell. Research*, 2005.

[11] G. Liu, T. Janhunen, and I. Niemelä. Answer set programming via mixed integer programming. *Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 32–42, 2012.

[12] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Computing*, 1(2):146–160, 1972.

[13] S. Thiébaux, J. Hoffmann, and B. Nebel. In defense of PDDL axioms. *Artificial Intelligence*, 168(1):38–69, 2005.

[14] T. Vossen, M. O. Ball, A. Lotem, and D. Nau. On the use of integer programming models in AI planning. In *Proc. IJCAI*, 1999.

[15] 三浦脩和. 汎用プランニングにおける Axiom 自動抽出について, 2016. 東京大学教養学部学際科学科卒業研究論文.

[16] 堀江慧. プランニング問題のドメイン非依存なモデリング手法:近年のソルバ高速化手法を考慮して, 2015. 東京大学教養学部学際科学科卒業研究論文.