Vol.2022-DPS-191 No.14
Vol.2022-MBL-103 No.14
Vol.2022-ITS-89 No.14
2022/5/26

時間割編成問題を記述可能なDSLとソルバとの連携方式

松田 陸斗[1]　乃村 能成[2]

概要：現在，少子化による大学再編の必要性や COVID-19 の感染拡大防止策による授業のオンライン化によって，大学の時間割を再編成したいという要求が増している．このため，大学の時間割作成システムの需要が高まっている．しかし，大学によって，時間割の制約が異なると考えられているため，時間割編成問題の表現方法の汎化が大きな課題である．我々は，時間割編成問題を記述可能な DSL である **AUK** を提案している．AUK が出力する AST に基づいて，時間割編成問題を解決するためのソルバを差換え可能である．本稿では，AUK とソルバを連携させるための手法について説明する．

キーワード：時間割編成問題，ドメイン特化言語，SAT ソルバ

1. はじめに

大学等の教育機関では，年度末に次年度の時間割編成作業が行われている．IT 化が進んでいる昨今においても，時間割編成作業は未だ手作業で行われることが多く，多大な労力を要することが知られている [1]．特に，大学の時間割編成は小中高校の時間割編成と比べて時間割の規模が大きく，時間割に課せられる制約の種類も豊富にあることから，時間割編成作業の労力が大きい．さらに昨今，少子化や COVID-19 の影響により，大学の授業形態が変わりつつあるため，時間割再編成の機会が増えている．このため，大学における時間割の自動編成システムが必要とされている．

これまでに，時間割自動編成システムに関する研究は数多く行われており [2] [3]，また，商用の時間割自動編成システムも販売されている [4] [5]．しかしながら，これらの中で広く実用化されているものはない．ここで，現状の時間割作成システムの課題を以下に示す．

(1) 大学の時間割編成問題を一般的に表現できる記法やインタフェースがない

(2) 一般的な大学の時間割編成問題の規模に対して，高品質な解を短時間で求解できない

課題 (1) について，大学によって独自の制約があると考えられるため，大学の時間割編成問題の一般化が困難である．また，時間割編成問題への関心の多くは解法についてであり，一般化についての議論は十分に行われていない．そこで，我々は時間割編成問題を記述可能なドメイン特化言語 (Domain Specific Language; DSL) を提案している [6]．本研究では，この DSL を時間割作成システムの入力インタフェースに用いることで課題解決を試みる．

課題 (2) について，一般的な大学の時間割編成問題の規模に対して，高品質な解を求めることは困難である．ここで，高品質な解とは，より多くの制約を満たす解のことである．現在，時間割編成問題については数多くの解法が提案されているものの，標準となっている解法はない [7]．本稿では，特定の時間割編成アルゴリズムに依存しない時間割作成システムを提案する．具体的には，本システムそのものは求解機能を有しておらず，外部の求解機能 (以下，ソルバと呼ぶ) に処理を委譲する設計を提案する．

本稿では，大学の時間割作成システムを提案し，本システムの構成，処理流れを述べる．本システムは，DSL を解析し，それをソルバが扱える形式に変換するコンパイラのような役割を担う．本システムを用いることで，時間割編成アルゴリズムの発展に伴い，不変な UI で求解性能の向上が実現できる．

以降では，2 章で時間割編成問題の概要と研究課題について述べる．3 章で提案する時間割作成システムの仕組みと処理流れを説明し，実際のデータを用いて DSL とソルバとの連携方式について述べる．

2. 時間割編成問題とその課題

2.1 時間割編成問題

時間割の編成は，時間枠，学生，教室，教員，授業のよ

[1] 岡山大学大学院自然科学研究科
[2] 岡山大学学術研究院自然科学学域

ⓒ 2022 Information Processing Society of Japan

Vol.2022-DPS-191 No.14
Vol.2022-MBL-103 No.14
Vol.2022-ITS-89 No.14
2022/5/26

うな時間割を構成する要素（以下，資源と呼ぶ）の組合せの決定である．つまり，幾通りも考えられる資源の組合せの中から，最適な組を決定する必要がある．通常，時間割編成の際には資源の組合せを制限するような制約が課せられる．したがって，時間割編成問題は，制約を満たすような資源の組合せを決定する問題といえる．

また, 時間割編成問題は世界的にも関心が高く, 競技会 [8] やカンファレンス [9] 等も開催されている．直近の競技会である ITC2019(International Timetabling Competition 2019) では，世界中の大学の時間割編成問題に対して，どれほど高品質な解を求められるかを競うものであった．さらに，ITC2019 では時間割編成問題の表現方法についても議論され, ITC2019 の競技用フォーマット（以下, ITC フォーマットと呼ぶ）が提案されている．ITC フォーマットは，世界中の大学の時間割編成問題が表現されていたことから，汎用性が高く，時間割編成問題を記述する DSL として有益な提案である．しかしながら，ITC フォーマットを時間割作成システムに取り入れた実例はなく，ITC フォーマット自体も記述性が高いとはいえない．そこで，我々は，ITC フォーマットと同等の記述力を持ち，かつ，記述性を向上させた DSL として，**AUK** を提案している [6]．本研究では，AUK を用いた時間割作成システムの提案を行う．

2.2 時間割編成問題における課題

本研究では，時間割編成問題における以下の課題についての対処方法を検討，提案する．

（1） 時間割編成問題を表現するフォーマットを定義する
時間割編成問題が抱える課題のひとつは，時間割編成問題の一般的な表現方法が規定されていないことである．この要因としては，多くの研究者の関心は時間割編成問題の求解方法や求解速度にあることや，大学毎に時間割が持つ性質が異なるため一般化が困難であることが挙げられる．本研究では，我々が提案している時間割編成問題を記述可能な DSL である AUK を本システムの入力インタフェースに用いることで，課題解決を試みる．

（2） 求解アルゴリズムを差換え可能なシステム構成にする
時間割編成問題では，多くの解法が提案されているが，未だ標準となっている解法はない．そこで，本システムでは時間割編成問題の中間表現を用いることで，様々なフォーマットへの変換可能な設計を実装する．これによって，求解アルゴリズムを差換え可能なシステム構成を実現する．なお，本研究では，時間割編成問題の解法そのものについての言及はしない．

（3） 時間割の修正作業を容易にする
時間割編成のユースケースとして，既存の時間割を修正する場合が考えられる．時間割の修正作業は，教員の要望等により連続的に行われたり，時間割の微調整のため反復的に行われたりすることが考えられる．しかし，制約が複雑に絡んでいる時間割の修正は容易でない．本研究では，DSL を用いることで対話的な修正を可能にする．時間割作成者は，DSL に記述されている制約を変更して再度時間割編成を行うことで，容易な時間割修正が可能になる．

3. 時間割作成システム

3.1 目的

時間割作成システムは，以下に示す二つの機能の連携を担っている．

（1） AUK
時間割編成問題を記述可能な DSL として，AUK の提案がある．AUK は，ITC フォーマットを基に提案されており，ITC フォーマットと比較してより簡素に記述可能である．

（2） ソルバ
本システムでは，様々な汎用的な求解ツールの差換えが可能である．現段階で，許容している求解ツールのひとつに SAT ソルバがある．SAT ソルバとは，充足可能性問題 (Satisfiability problem; SAT) を解くツールである．時間割編成問題は SAT に定式化可能であるため，本稿では SAT ソルバを用いてシステムの処理流れを説明する．

本章では，提案する時間割作成システムの構成と処理流れについて説明し，DSL とソルバとの連携について説明する．

3.2 時間割作成システムの構成

時間割作成システムの構成図を**図 1** に示す．本システムは，AUK で記述された入力を AUK Parser によって抽象構文木 (Abstract Syntax Tree; AST) に変換し，Converter を用いて AST から様々な形式に変換することでデータの流通が行われる．Converter には，AST をソルバが扱える形式に整形する Solver Converter の他に，HTML，CSV，AUK に対応する Converter がある．

本システムでは，AST のような中間表現を用いることで，時間割データの様々な形式への変換を容易にしている．例として，時間割編成問題を解くためにソルバの入力形式に変換する場合や，求解結果としての AST を人が見やすい形式に変換する場合が考えられる．

3.3 時間割作成システムの処理流れ

本節では，提案する時間割作成システムが AUK で記述

ⓒ 2022 Information Processing Society of Japan

Vol.2022-DPS-191 No.14
Vol.2022-MBL-103 No.14
Vol.2022-ITS-89 No.14
2022/5/26

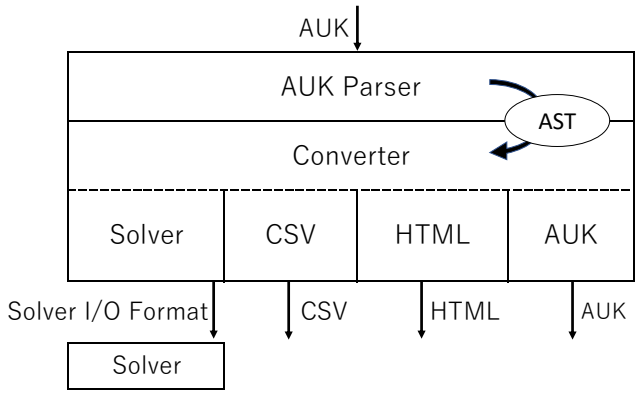

図 1 時間割作成システムの構成図

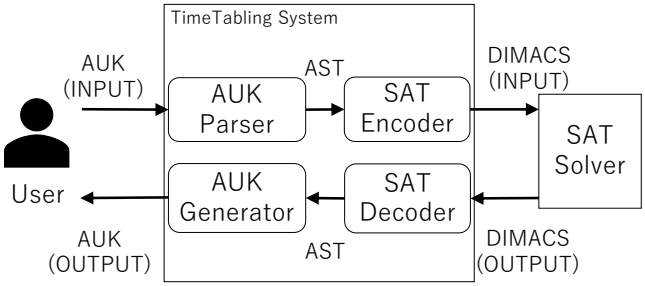

図 2 時間割作成システムの処理流れ

された資源，および制約を DIMACS フォーマット [10] に変換し，SAT ソルバを用いて求解することで資源の自動割当てを行う処理流れを説明する．ここで，DIMACS フォーマットとは，SAT ソルバの標準入出力フォーマットである．

本システムは，必要な入力情報を AUK で渡されることを想定している．また，作成した時間割の情報も AUK で出力される．これによって，時間割作成者は生成された AUK を編集することで，対話的な時間割修正が可能になる．

ここで，提案する時間割作成システムの構成要素とデータ流れを**図 2** に示す．

図 2 を中心に，実際の時間割作成の処理流れを以下で説明する．

(1) ユーザは時間割資源，および時間割制約を AUK で記述し，時間割作成システムに入力する．

(2) 時間割作成システム（AUK Parser）は，AUK を解析し，AST を生成する．ここで，AST とは，AUK と DIMACS フォーマットの中間表現である．

(3) 時間割作成システム（SAT Encoder）は，AST から SAT 符号化を行い，DIMACS フォーマットを生成する．

(4) SAT ソルバは，与えられた SAT の充足可能性判定を行い，結果を DIMACS フォーマットで生成する．

(5) 時間割作成システム（SAT Decoder）は，DIMACS フォーマットを解析し，AST を生成する．

(6) 時間割作成システム（AUK Generator）は，AST から AUK を生成する．

表 1 考慮する資源一覧

時間枠	月曜 1 限
	月曜 2 限
	…
	金曜 4 限
教室	講義室 1
	演習室 1
教員	佐藤先生
	林先生
授業	計算機理論
	プログラミング演習

3.4 DSL と時間割作成システム間の連携

3.4.1 AUK

AUK とは，時間割編成問題を記述可能な DSL である．AUK は，ITC フォーマットと同等の記述力で，より簡素に記述できる．ここでは，時間割編成問題に対する AUK の記述例を示す．

まず，考慮する資源を**表 1** に示す．例における資源は，時間枠 (曜日 : 5) × (時限 : 4) = 20 通り，教室 2 通り，教員 2 通り，および授業 2 通りを考慮する．

次に，考慮する制約を以下に示す．

(1) 「演習室 1」は 3 限以降利用不可能

(2) 「林先生」は月曜日のみ出勤可能

(3) 「プログラミング演習」は「演習室 1」で開講可能

(4) 「プログラミング演習」は「林先生」が担当可能

(5) 「計算機理論」の後に連続して「プログラミング演習」が開講される

ここで，時間割の制約は，大きく 2 種類に大別できる．制約 (1)(2)(3)(4) のように，異なる資源間の組合せを制限する制約をドメイン制約と呼ぶ．また，制約 (5) のように，同じ資源間の競合に関する制約を競合制約と呼ぶ．

例示した資源および制約を AUK で記述したものを**リスト 1** に示す．

AUK は，資源および制約を宣言的に記述できる．ドメイン制約は，制約が課せられる資源内に記述する．一方，競合制約は，制約の内容に相当する予約語を記述する．たとえば，制約 (5) は，35 行目の `NextTime` が対応している．

3.4.2 AUK Parser

本システムでは，AUK で記述された時間割の情報を

ⓒ 2022 Information Processing Society of Japan

Vol.2022-DPS-191 No.14
Vol.2022-MBL-103 No.14
Vol.2022-ITS-89 No.14
2022/5/26

リスト 1　AUK の記述例

```
initialize do
  nr_days_a_week 5
  nr_periods 4
end

period do
  first start_time:"8:00", end_time:"9:50"
  second start_time:"10:00", end_time:"11:50"
  third start_time:"13:00", end_time:"14:50"
  fourth start_time:"15:00", end_time:"16:50"
end

room "講義室1" do
end

room "演習室1" do
  unavailable start_time:"13:00", end_time:"16:50"
end

instructor "佐藤先生" do
end

instructor "林先生" do
  unavailable every: ["Tue", "Wed", "Thu", "Fri"]
end

lecture "計算機理論" do
end

lecture "プログラミング演習" do
  rooms "演習室1"
  instructors "林先生"
end

NextTime do
  lectures "計算機理論", "プログラミング演習"
end
```

AUK Parser で解析して，システム内で AST オブジェクトとして保持する．ここで，AST オブジェクトの構造図を**図 3** に示す．図 3 は，リスト 1 で示した AUK の記述例を AST オブジェクトに変換したものである．AST オブジェクトが持つ情報は，大きく分けると資源と競合制約に分類される．資源は，時間枠，教室，教員，および授業の 4 種類から構成され，各資源はドメイン制約を持っている．競合制約は，同一資源間にかかる制約であり，主に授業間にかかる制約を表している．

3.4.3　AUK Generator

AUK は，時間割作成システムのインタフェースとしての利用用途以外にも，AST を永続化させるための形式としても利用できる．たとえば，時間割の修正を行いたい場合，一度 AST を AUK に変換して，AUK の記述を変更し，再度 AUK Parser に渡すことで容易に修正可能となる．AUK Generator は，AST から AUK を出力する．

3.5　SAT ソルバと時間割作成システム間の連携

3.5.1　SAT ソルバ

本システムでは，時間割編成問題の求解機能の一つとして，SAT ソルバを使用可能である．SAT ソルバとは，充足可能性問題を解くツールである．SAT ソルバは，連言標準形の充足可能性問題（CNF-SAT）を入力として受け取り，求解を行う．求解結果が充足可能であった場合，充足する変数の割当てを出力する．ほとんどの SAT ソルバは，DIMACS フォーマットに対応しているため，ソルバ自体は容易に差換え可能である．

3.5.2　DIMACS フォーマット

DIMACS フォーマットは，充足可能性問題を表現する標準フォーマットであり，SAT ソルバの入出力フォーマットでもある．DIMACS フォーマットの例を**リスト 2** に示す．

リスト 2　DIMACS フォーマットの記述例

```
p cnf 5 3
1 -5 4 0
-1 5 3 4 0
-3 -4 0
```

リスト 2 は，命題論理式の連言標準形を記述している．p からはじまる行は，DIMACS フォーマットの宣言部分である．それぞれ，形式は cnf，命題変数は 5 個，節数は 3 個であることを表している．リスト 2 の例は，命題変数 p_1, p_2, p_3, p_4, p_5 について，以下の論理式を表している．

$$(p_1 \vee \neg p_5 \vee p_4) \wedge (\neg p_1 \vee p_5 \vee p_3 \vee p_4) \wedge (\neg p_3 \vee \neg p_4)$$

また，リスト 2 を SAT ソルバに渡したときの出力形式を**リスト 3** に示す．

リスト 3　リスト 2 を SAT ソルバに渡した結果（DIMACS フォーマットの出力形式）

```
SAT
-1 -2 -3 -4 -5 0
```

DIMACS フォーマットの出力は，すべての命題変数について真の場合は符号なしで，偽の場合はマイナス符号が付いて出力される．リスト 3 の場合，すべての命題変数に偽が割当てられた場合に充足可能であることを示している．

3.5.3　SAT Encoder

SAT Encoder は，時間割作成システム内に保持されている AST を DIMACS フォーマットに変換する処理を行う．以降では，図 3 の AST から CNF-SAT への定式化の処理流れを示し，最終的に出力される DIMACS フォーマットを例示する．ここで，AST から CNF-SAT への定式化の

ⓒ 2022 Information Processing Society of Japan

Vol.2022-DPS-191 No.14
Vol.2022-MBL-103 No.14
Vol.2022-ITS-89 No.14
2022/5/26

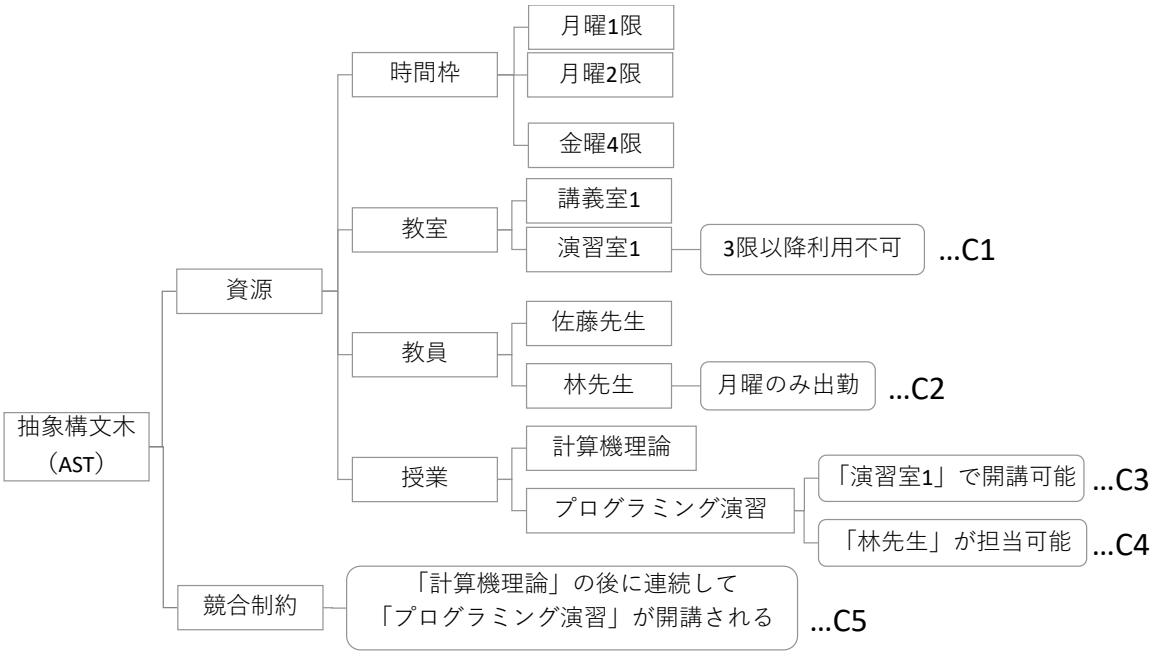

図 3 リスト 1 を変換した AST 構造図

大まかな処理流れを以下に示す．

(1) 命題変数を定義する

(2) 命題変数の有効範囲を定める

(3) 命題論理式を生成する

3.5.4 命題変数を定義する

ひとつの命題変数は，「授業 X は，時間枠 X に教室 X で教員 X によって開講される」という命題を表す．したがって，定義される命題変数の数は各資源の組合せ数と同じになる．定義される命題変数一覧を**表 2** に示す．この場合の命題変数の数は，(授業数 : 2) × (教員数 : 2) × (教室数 : 2) × (時間枠数 : 20) = 160 個 である．

3.5.5 命題変数の有効範囲を定める

資源には，ドメイン制約が定義されている場合がある．ドメイン制約に違反する命題は必ず偽になり，このような命題を無効な命題とする．SAT Encoder では，事前に無効な命題を削除している．図 3 の `C3`，`C4` に示すドメイン制約によって，「プログラミング演習」が「講義室 1」で開講される命題 (p_{41}〜p_{60})，および「プログラミング演習」が「佐藤先生」によって担当される命題 (p_1〜p_{40}) を削除する．さらに，図 3 の `C1`，`C2` より「プログラミング演習」が開講可能な時間枠は「月曜 1 限」，および「月曜 2 限」の 2 通りになるように枝刈りされる．図 2 からドメイン制約による枝刈りをした後の命題変数一覧を**表 3** に示す．

3.5.6 命題論理式を生成する

定義された命題変数を用いて，競合制約を CNF-SAT に定式化する．例として，図 3 の `C5` に示す競合制約 “「計算機理論」の後に連続して「プログラミング演習」が開講される” の定式化を行う．授業「計算機理論」が時間枠 t に開講される命題変数を p_{ct}，授業「プログラミング演習」が時間枠 t に開講される命題変数を p_{pt} とすると，上記の競合制約は以下のように定式化できる．

$$p_{ct} \rightarrow p_{p(t+1)} \tag{1}$$

ただし，t+1 は時間枠 t の後に連続する時間枠を表す．これを選言節に変換すると以下の様になる．

$$\neg p_{ct} \vee p_{p(t+1)} \tag{2}$$

したがって，「計算機理論」が時間枠 t に開講される命題変数と，「プログラミング演習」が時間枠 $t+1$ に開講される命題変数のすべての組合せに対して，式 (2) を適用することで定式化できる．これによって，生成される DIMACS フォーマットの例を**リスト 4** に示す．

リスト 4 SAT Encoder によって最終的に生成される DIMACS フォーマット

```
1 p cnf 82 4
2 -81 62 0
3 -101 62 0
4 -121 62 0
5 -141 62 0
```

また，この CNF-SAT の充足結果を**表 4** に示す．表 4 より，真が割り当てられた命題変数は p_{62} と p_{81} の二つのみであった．つまり，この 2 通りの資源の組合せが，すべての制約を満たす資源の組合せであり，実行可能な時間割である．

ⓒ 2022 Information Processing Society of Japan

Vol.2022-DPS-191 No.14
Vol.2022-MBL-103 No.14
Vol.2022-ITS-89 No.14
2022/5/26

表 2　定義される命題変数一覧

命題変数	授業	教員	教室	時間枠
p_1	プロ演習	佐藤先生	講義室 1	月曜 1 限
p_2	…	佐藤先生	講義室 1	月曜 2 限
p_3	…	佐藤先生	講義室 1	月曜 3 限
p_4	…	佐藤先生	講義室 1	月曜 4 限
p_5	…	佐藤先生	講義室 1	火曜 1 限
p_6	…	佐藤先生	講義室 1	火曜 2 限
…	…	…	…	…
p_{21}	…	佐藤先生	演習室 1	月曜 1 限
p_{22}	…	佐藤先生	演習室 1	月曜 2 限
…	…	…	…	…
p_{41}	…	林先生	講義室 1	月曜 1 限
…	…	…	…	…
p_{81}	計算機理論	佐藤先生	講義室 1	月曜 1 限
…	…	…	…	…
p_{160}	…	…	…	…

表 3　ドメイン制約による枝刈り後の命題変数一覧

命題変数	授業	教員	教員	時間枠
p_{61}	プロ演習	林先生	演習室 1	月曜 1 限
p_{62}	プロ演習	林先生	演習室 1	月曜 2 限
p_{81}	計算機理論	佐藤先生	講義室 1	月曜 1 限
…	…	…	…	…
p_{160}	…	…	…	…

表 4　SAT ソルバによる求解後の充足結果として真が割当てられた命題変数一覧

命題変数	授業	教員	教室	時間枠
p_{62}	プロ演習	林先生	演習室 1	月曜 2 限
p_{81}	計算機理論	佐藤先生	講義室 1	月曜 1 限

3.5.7　SAT Decoder

前節で，実行可能な時間割が得られたが，実際の情報は命題変数の真理値にすぎないため，これを解釈する必要がある．SAT Decoder は，SAT ソルバの充足結果を受け取り，これを AST に反映する．

表 4 の充足結果について，実際に得られる DIMACS フォーマットを**リスト 5** に示す．

リスト 5　SAT ソルバによる求解後の充足結果を表す DIMACS フォーマット

```
1 SAT
2 -61 62 81 -82 ... -160 0
```

リスト 5 より，真が割り当てられた命題変数は，62($= p_{62}$) と 81($= p_{81}$) の二つのみであった．

SAT Decoder は，リスト 5 を復号し，意味を解釈して AST に反映させる．その後，AUK Generator によって編成された時間割が AUK で出力され，システムの処理は終了する．最終的に出力される AUK を**リスト 6** に示す．

リスト 6　編成した時間割を AUK Generator で出力した結果 (一部)

```
27 lecture "計算機理論" do
28   room "講義室1"
29   instructor "佐藤先生"
30   period "Mon1"
31 end
32
33 lecture "プログラミング演習" do
34   room "演習室1"
35   instructor "林先生"
36   period "Mon2"
37 end
```

リスト 6 は資源の組合せを表しており，リスト 1 の 27〜33 行目の `lecture` 節に結果が挿入される形で出力される．AUK の出力形式は，入力形式と比べて `lecture` 内のドメイン制約の指定子が単数形になっている．これは，単数形の指定子は割り当てが確定した資源，複数形は割り当ての候補となる資源を表している．修正作業は，必要に応じて `lecture` 内の指定子を複数形に編集して，再度時間割作成システムに渡すことで可能となる．

4.　関連研究

時間割作成システムの提案として，汎用プロジェクトスケジューラ SEES を大学の時間割作成に適用した例がある [2] [3]．文献 [2] では，SEES を用いて約 0.37 秒で制約を満たす許容解を得られたことを示すとともに，時間割作成システムのインタフェースの重要性，困難性について述べられている．また，文献 [3] では，時間割作成システムを用いて作成した時間割が，実際に大学で運用された例を紹介している．

時間割作成システムに関する研究の多くは，具体的な時間割編成問題を解決するシステムの提案である．一方，我々の研究は，ITC フォーマットのような汎用的なフォーマットを基にして，多くの時間割に対する要求を表現できる形から始めている．このように，ある程度汎用化されたフォーマットから，具体的な時間割編成問題を表現できるように落し込んでいくことで，システムの汎用性の向上を試みる．

また，時間割編成問題の求解アルゴリズムの提案としては，最適化問題のメタヒューリスティクスアルゴリズムが適用される例がある [11]．文献 [11] では，時間割編成問題を例に組合せ最適化問題のメタヒューリスティクスの比較

ⓒ 2022 Information Processing Society of Japan

を行い，著者らが開発した大学時間割編成問題に特化したアルゴリズムである MAX-MIN Ant System が最も良い性能を示した．ITC2019 の議論においても，複雑な時間割作成にメタヒューリスティクスを用いることが有用であることが分かっている．

最適化アルゴリズムを本システムに取り入れる場合, AUK の改良，および最適化アルゴリズムの Solver Converter が必要となる．時間割編成問題を最適化問題へ定式化する場合，制約はその重みによって以下の 2 種類に分類される．

(1) ハード制約: 必ず満たさなければならない制約

(2) ソフト制約: 可能なら満たしたい制約

最適化問題では，すべてのハード制約を満たしつつ，より多くのソフト制約を満たす解を求める．したがって，AUK で記述する制約についても，ハード制約かソフト制約を指定できるような文法が必要になる．また，本システムはソルバを利用して求解を行うため，最適化アルゴリズムの API への Converter の実装が必要になる．

5. おわりに

本稿では，大学の時間割作成システムを提案し，時間割編成の処理流れを述べた．本システムは，時間割の自動編成に関する二つの課題の解決を試みるものである．一つ目の課題は，時間割作成システムの統一的なインタフェースの実現である．これについては，時間割編成問題を記述可能な DSL を提案し，システムのインタフェースとして実装した．二つ目の課題は，大学の時間割編成問題に対する高速，高品質な求解である．これについては，本研究では言及していない．代わりに，特定の解法に依存しないシステム設計を提案している．

今後の課題として，提案したシステムの実装，および実際の大学時間割の作成が挙げられる．本システムを用いて実際の時間割を作成することで，システムの評価，および改良を試みる．

参考文献

[1] 池上敦子，呉　偉：学校時間割作成，日本オペレーションズ・リサーチ学会 (2020).
[2] 堀尾正典：汎用プロジェクトスケジューラの大学時間割問題への適用，名古屋学芸大学 教養・研究紀要， Vol. 4, pp. 61–74 (2008).
[3] 堀尾正典：実用的な大学時間割作成システムの開発，日本経営工学会論文誌， Vol. 62, No. 1, pp. 1–11 (2011).
[4] クラウド版 校務システム CampusForce：時間割作成ソフト Koma Force，ネットフォース株式会社（オンライン），入手先 ⟨https://www.campusforce.net/koma/⟩（参照 2022-03-14）.
[5] HEURIS 校務支援：時間割ソフト YELL，株式会社 管理工学研究所（オンライン），入手先 ⟨https://www.kthree.co.jp/heuris/yell/index.html⟩（参照 2022-03-14）.
[6] 松田陸斗，乃村能成：大学の時間割編成問題を記述可能な DSL と時間割作成システムの提案 (2021).
[7] Chen, M. C., Sze, S. N., Goh, S. L., Sabar, N. R. and Kendall, G.: A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities, Vol. 9, pp. 106515–106529 (2021).
[8] Müller, T., Rudová, H. and Müllerová, Z.: ITC2019: International Timetabling Competition 2019, (online), available from ⟨https://www.itc2019.org⟩ (accessed 2021-01-13).
[9] Ozcan, E.: PATAT - International Conference on the Practice and Theory of Automated Timetabling, PATAT (online), available from ⟨http://patatconference.org/⟩ (accessed 2022-03-17).
[10] Matti, J., Ärvisalo, Olivier, R., Oussel and organizers: Rules of the 2011 SAT Competition (2011).
[11] Socha, K., Knowles, J. and Sampels, M.: A MAX-MIN Ant System for the University Course Timetabling Problem, pp. 1–13 (2002).

ⓒ 2022 Information Processing Society of Japan