4K-04

Z3 を用いた時間割及び履修制限の妥当性検査支援の提案

佐々木　遼加†　　五百蔵　重典‡

神奈川工科大学情報学部情報工学科‡

1. はじめに

大学の履修要綱には，配当科目の選必および卒業要件だけでなく，卒業研究着手要件や各科目の履修のための条件など，細かい条件が記載されている．学生は，このルールに則って科目を履修し，卒業を目指す．教員はこれらの履修要綱に書かれた配当科目を開講し，時間割として提示する．時間割は，同一学年の必修科目を同じ時限に配当しないようにする必要があるだけではなく，単位を落として再履修する上位学年の学生も考慮する必要がある．本来の必修科目および再履修科目が同じ時限に配当されていると，片方しか履修できないためである．そして場合によっては，4 年で卒業できない学生を大量に生んでしまう可能性がある．上記の理由から，履修要綱に書かれている規則および時間割は，相互に密接に絡み，慎重に作成する必要がある．

これらの検査を人手ではなく，プログラムで検査することは有用であるが，頻繁に変わりうる履修要綱のたびに，プログラムを正しく作成するのは手間である．そこで，本研究では，近年注目が高まってきている SMT Solver[1]を用いて，これら履修条件と時間割を制約として記入し，進級・卒業の可否を判定することを試行する．これにより，現実的な問題である本問題の記述量がどの程度になるのか，検査にかかる時間はどの程度であるかを示す．さらに，与えられた制約を変更して，再履修が即留年となる科目はあるか，再履修可能な科目はいくつあるのかなどを，確認することを試みる．再履修可能な科目が多い時間割は，柔軟性が高い時間割とみなせるため，これらを確認できることは意義があると考えている．

2. 検証手法

本研究では，Microsoft が開発した SMT Solver である Z3 を使う．SMT Solver は，整数を含む論理式の充足可能問題を解くことができる．そこで，これら履修要件および時間割を論理式として記述し，充足可能であることを確認することで，4 年間で卒業可能であるなどの判定を行うことが可能である．

本研究では，Z3 をより扱いやすくした Python モジュールである Z3py を利用して，充足可能問題を扱う．

履修要綱の情報は，科目名，単位数，配当時間などを表形式(csv 形式)で格納する．この時，科目名で番号がある場合は昇順で格納しておくものとする．

本研究では，本学の実際の履修要綱および時間割を用いて，(1) 卒業要件，(2)卒業研究着手要件，(3) 時間割をそれぞれ検証する．また本検証では，本研究と同様に実際の例を扱った片山[3]を参考にし，これらを別々のプログラムとして実装をした．理由として，それぞれで使用する条件が異なり，それぞれ独立した変数を与えるため，別々のプログラムで処理したほうが，効率が良いと考えたためである．

以下に，制約の記述方法の考え方の概略をいくつか示す．ただし，以下の記述の s は，Z3py 上の Solver を表すオブジェクト名である．

履修制限を確認するモデルを作成するために，Z3py の変数に，未履修は 0，１年前期に取得ならば１，１年後期に取得なら 2 のように，単位取得時期を数値が制約として割り当てられるようにする．例えば科目に対応する変数名（以下，科目変数）を Gairon とし，この科目変数が表す科目が，2 年次の配当で，選択科目である場合は Z3Py の機能である論理和 Or を使い論理和を表現し，add を用いて「s.add(Or(Gairon == 0, 2 <= Gairon))」のように制約を記述する．

対象とする科目数が多いため，Z3py の Intvector という機能で配列を用意し，科目変数に対応している単位取得時期を配列に格納するという工夫を行った．これにより，for 文を用いて簡単に制約を指定できるようになる．例えば英語の履修制限を確認する場合，科目変数は Ce であり，作成した配列は Eng とする．この時，英語

Proposal of Validity Check Support for Timetable and Course Restrictions using Z3
†Haruka Sasaki
‡Shigenori Ioroi
‡Kanagawa Institute of Technology

Ⅱを履修するためには英語Ⅰを履修していなければならない場合は「Eng[0] < Eng[1]」という制約を与えて，制約を満たした充足があるかを確認できる．

同一の時間帯に配当されている授業を同一学年では履修できないことを示すために，以下のように記述する．例えば，C 言語 2(変数 CLang2)は C 言語 1(変数 CLang1)が同じ時間に配当されている場合，CLang1 と CLang2 には同じ値（同一開講時間帯の履修）を割り当てられない．この制約を示すために，Z3 の機能である Distinct を用いて「s.add(Distinct(CLang1, CLang2))」と記述する．

3. 検証

本検証では 2019 年度入学生用の履修要綱を例題とする．

卒業要件を検証する．教育区分の系統別に科目変数を割り当て，「科目変数，科目名，必修/選択，単位数，履修/未履修，履修時期」を 1 行のデータとする csv 形式を用いて検証を行った．本検証の条件の一部は以下のようになる．

- 総単位数 124 単位以上取得・・・・・①
- 人文社会系 a 群 4 単位以上取得・・・②
- 共通基盤 32 単位以上取得 ・・・・・③
- 専門 75 単位以上取得 ・・・・・・・④
- 任意単位は 17 単位以上取得 ・・・・⑤

である．それぞれをプログラム上で表現すると，①は変数 kei を用意し「s.add(kei >= 124)」となる．②は人文社会学で取得した単位数を格納されている配列 data_cha の合計が，4 単位以上であるという条件は「s.add(Sum(data_cha) >= 4)」となる．③,④は変数 kyotu,sen を用意し「s.add(kyotu >= 32)」，「s.add(sen >= 75)」となる．⑤の条件は，①，③および④の条件を満たすとき，必ず真になる．そのため，制約として記述する必要がない．上記以外の制約も記述すると，科目数が 71，制約の個数が 56，変数(配列含む)は 24 個となり，検証に要する時間は約 3.6 秒であった．

卒業研究着手要件を検査する．卒業研究着手要件として，条件分類ごとに新たに科目変数を割り当て，「科目変数，科目名，履修/未履修」を 1 行とする csv 形式を使用した．またこのデータの履修/未履修のデータは履修済みを 1，未履修は 0 が格納されている．さらに 3 年次終了時の単位数計算のために卒業要件で使用した csv 形式を用いて検証を行った．本検証の条件の一部は以下のようになる．

- 3 年次終了時総単位 104 単位以上 ・・・・①
- 3 年次終了時共通基盤単位 26 単位以上・・②
- 指定された 8 科目から 7 科目以上の取得・③
- 指定された 7 科目すべての取得・・・④

である．それぞれをプログラム上で表現すると，①および②はそれぞれ変数 kei3，kyotu3 を用意し，各々「s.add(kei3 >= 104)」，「s.add(kyotu3 >= 26)」となる．③は指定された 8 科目の科目変数を kiso とし，これらの履修済みか未履修かを配列に格納し，Sum を用いて制約を表現すると「s.add(kiso == Sum(data_kiso))」,「s.add(kiso >= 7)」となる．④は科目変数が 0 でないことを示せばよい．上記以外の制約も記述すると，科目数が 92，制約の個数が 51，変数(配列含む)は 16 個となり，検証に要する時間は約 3.4 秒であった．

時間割配当要件として，本検証では必修のみの時間割の条件を考えた．なお，配当学年，開講時期はあらかじめ設定した．本検証での条件は

- 必修同士が被らない ・・・・・・・・・①
- 前期（後期）は全部で 20 コマある・・②

になる．①は Distinct を用いて記述でき，「s.add(Distinct(data_f))」となる．②は「s.add(And(data_f[i] >= 1,data_f[i] <= 20))」で表現できる．これらの制約を実行したところ，科目数が 20，制約の個数が 3，変数(配列含む)は 2 個で，検証に要する時間が約 3.1 秒であった．

4. まとめと今後の課題

本研究では，大学の履修要件と授業配当を，手続き的に検査するのではなく，制約による記述が可能なことを示した．検査にかかった時間も許容範囲であることを確認した．

制約による記述は，与えられている条件をほぼそのまま制約に置き換えられるため，可読性も高いことが分かった．

今後の課題として，時間割の検証では必修のみの検査となってしまったため，選択科目も含めた時間割の検証を進めたい．さらに，必修が重なっていても 4 年で卒業できる時間割の配当の検証を進めていきたいと考えている．

参考文献

[1] 宋剛秀，番原陸則，田村直之，鍋島英知：SAT ソルバーの最新動向と利用技術，コンピュータソフトウェア，Vol.35,No4,pp72-92,2018

[2] 梅村晃広：SAT ソルバ・SMT ソルバの技術と応用，コンピュータソフトウェア，Vol.27,No3，pp24-35,2010

[3] 片山卓也：国民年金法の述語論理による記述と検証 SMT ソルバーZ3Py を用いたケーススタディ，コンピュータソフトウェア，Vol.36,No.3,pp33-46,2019