Title	法令工学の実践　国民年金法の述語論理による記述と検証
Author(s)	片山，卓也
Citation	
Issue Date	2023-7
Type	Book
Text version	publisher
URL	http://hdl.handle.net/10119/16096
Rights	片山卓也．法令工学の実践　国民年金法の述語論理による記述と検証．第5版，JAIST Press，2023，146p．
	2019年10月 第1版発行 / 2019年11月 第2版発行 / 2020年11月 第3版発行 / 2021年12月 第4版発行 / 2023年7月 第5版発行 / ISBN:978-4-903092-53-9，2023年7月5日 第5版公開，[第１版からの変更] 第十四条年金原簿：変更，第九十四条保険料の追納：追加，[第2版からの変更] 変更，修正(第十四条　年金原簿，第八条　資格取得の時期，第九条　資格喪失の時期，年金原簿の記述法)，述語名の変更(第二十六条，第二十八条)，追加（第九十四条　保険料の追納，附則（平成一六年六月一一日法律第一〇四号）第十九条)，追加(実際の暦による日付けの記述)，追加(論理式記述に基づく年金システムシミュレーターの構成の概略)，[第３版からの主な変更]・条文論理式において他の条文や年金原簿で定義された述語をインポートし参照する形式を「述語名」から{「条文番号」，「年金原簿名」}.「述語名」に変更した．これにより，条文間の関連が明確となると同時に，年金原簿の操作の前後における変更の記述が明確となった．・検証で必要とする年金原簿，基本用語定義集の設定

Description	内容を検証スクリプト内に直接記述する方式に変更した．・基本用語定義集definitions,definitions_date において，事由成立の前後関係オペレータの名称と関数性を変更した．充足リスト，月数定義アルゴリズムの最適化を行った．，[第４版からの主な変更]・用語定義集definitions_date_3 における通日⇔西暦年月日変換において多用される実数除算＋floor 関数の実現が大井によるquot 関数により極めて簡単に実現できることが判明し[4]，これまでの量限定子を用いる方法に代えてこれを採用した．　・期間，月数などの日付オペレータについては，量限定子を用いた論理式による定義に先だって，充足性判定で行うPython 版を優先して使用する形にした．　・条文論理式の表現が条文の文章としても読み下せることを考え，他の条文で定義される述語の参照の方式を，条文番号・述語名の形式から述語名の形式に戻した．　・「論理式記述に基づく年金システムシミュレーターの構成」の内容を改めた．　・第八条，第九条のノートの記述を改めた．

法令工学の実践
国民年金法の述語論理による記述と検証

片山 卓也
katayama@jaist.ac.jp

第 5 版 (2023 年 7 月)
JAIST Press

概要

国民年金法のような行政サービスに関する法令は，その内容は明確であり，文章上の見かけの複雑さの割には論理的深度は深くない．したがって，論理式などの形式的体系によってその内容を記述することにより，可読性が高く，機械的なテストや分析が可能な法令記述が得られる可能性が高い．このような法令は我々の社会の制度的基盤であり，その実働化 IT システムが我々の社会生活を支えていることを考えると，法令に対する形式的技術の確立は非常に重要である．

本書は，このような立場から国民年金法の基本的条文の述語論理による記述と定理自動証明器（SMT ソルバー）Z3Py による検証について述べたものである．近年の定理証明技術の進歩や計算機システムの高性能化により，このような技術は行政サービスを規定する法令の作成や管理のための基盤技術になりえるものと考える．

また，定理自動証明器は条文論理式の実行機構という側面を持ち，法令実動化 IT システムの基本アーキテクチャと考えることができる．度重なる法令改正への対応により現在の法令実働化 IT システムの内部構造の劣化が進み，保守の困難性が社会的な問題となっているが，本書で紹介する法令の形式化・検証技術はこの問題に対する有力な解決法になると期待できる．

本書は 2 部から構成されている．

第 I 部はこのような新しい法令作成方法論の原理的可能性を示すために書かれた，日本ソフトウェア科学会誌「コンピュータソフトウェア」36 巻 3 号 (2019) に掲載された論文をそのままの形で転載したものである．

第 II 部は，第 I 部で述べた内容を事例の面から補強するために，そこで紹介した方法を国民年金法のより多くの条文へ適用した結果を詳細かつ具体的に述べたものであり，このような手法が広く実践される助けとなることを目的としたものである．各条文ごとに，(1) 条文の文章，(2) 論理式記述，(3) 検証スクリプトと検証結果，(4) それらの内容に関するメモを掲載した．

論理式記述は，マクロ表現を用いてなるべく簡潔になるように心懸けたが，これらのマクロの定義などを基本用語定義集として纏めた．これには日付や期間，月数，年齢，イベント成立の前後関係などが含まれ，年金法に内在する時間の構造を表現している．

一方，検証においてはテストデータとしての年金原簿が必要であるが，各条文の検証に必要な複数の年金原簿も纏めて収録した．また，日付けを代数的抽象データとして扱った検証事例や論理式記述に基づく年金システムシミュレーターの構成の概略なども収録した．

最後に，筆者は年金サービスに関する専門家ではなく，本書の論理式化は筆者が注釈書 [1, 2] などを頼りに条文を理解することにより行ったものである．この点において専門家から見た場合には条文の理解に不備がある可能性はあるが，最終的な結論である行政サービスに関する法令の論理式化と形式検証の可能性については，大きな間違いはないと考えている．この点も含め，内容に関する疑義に関してはご指摘いただければ幸いである．

第 4 版からの主な変更点は以下である．

- 用語定義集 `definitions_date_3` における通日⇔西暦年月日変換において多用される実数除算＋ floor 関数の実現が大井による quot 関数により極めて簡単に実現できることが判明し [4]，これまでの量限定子を用いる方法に代えてこれを採用した．これにより形式化の簡単化と実行時間の大幅な短縮が可能となった．
- **期間，月数**などの日付オペレータについては，量限定子を用いた論理式による定義に先だって，充足性判定で行う Python 版を優先して使用する形にした．これにより，大きなデータに対する検証を許容できる速度で実行出来るようになった．
- 条文論理式の表現が条文の文章としても読み下せることを考え，他の条文で定義される述語の参照の方式を，**条文番号・述語名**の形式から**述語名**の形式に戻した．
- 「論理式記述に基づく年金システムシミュレーターの構成」の内容を改めた．
- 第八条，第九条のノートの記述を改めた．

目次

第 I 部

原理編
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

(日本ソフトウェア科学会誌「コンピュータソフトウェア」Vol.36, No.3 (2019), pp.33-46 からの転載)

1　はじめに

我々の社会は多数の法令によりその構造や活動が定められている．このような法令が社会の要請や立法者の意図に沿った形で正しくつくられ，誤りなどがないことは社会の最も基本的な安心基盤である．これまで法令の作成は伝統的に人手により行われて来たが，近年法令が大量に作られるようになり，従来の職人的技量に頼るだけではその品質の維持が困難になりつつあり [1][2]，新しい時代の要請を満たす法令作成方法論の導入が求められている．その一方で法令の内容を具現化する情報システム（例えば，年金機構の年金情報システム）の開発においては，大規模で複雑なソフトウェアを正しく構築し維持する技術が長年にわたり研究され実践されてきたが，このような情報技術は法令自身の作成に於いても有効であると考えられる．

このような観点から“法令工学”の提案が片山等により行われている [3][4]．ソフトウェアの開発では，要求分析，設計，コーディング，テスト，検証や再利用技術などが，それらを支えるツール群とともに日常的に使われているが，これらの技術は法令の作成にも適用できるはずである．また，角田は法令の作り方とソフトウェア開発の間の類似性について指摘すると同時に，条例の再利用開発という観点から高度な検索機能をもつ条例データベースの開発を行っている [5][6]．

法令に情報システム開発技術を適用する際の最大の問題は，その非形式性にある．法令文は文章の構成や用語の選択などに関して，通常の文書に比べると格段に注意深くまた明確に作られているが，計算機を利用してその論理的処理を行えるようにはなっていない．したがって，法令の誤りや不整合の検出には，現状では，人間の目視に頼らざるを得ない．この観点から島津は，法文の構造的書き換えを提案し国民年金法に適用した [7]．これは法文の可読性を上げる上では非常に有効な方法であるが，計算機プログラムに対して行われるような機械的かつ網羅的なテストや検証を行う事は期待できない．このようなテストや検証を適用するには，何らかの方法で法令を形式化し，定理証明プログラムなどにより，その実行や推論を行えるようにすることが必要である．

法令の形式化の一つとして，述語論理による記述は古くから計算機科学者の興味を引き研究が行われて来たが [9]，多くのものは法律エキスパートシステムの構築との関連において行われ [10]，論理式からの法的推論を目的とするものである．法的推論は，解釈の幅により条文だけではその意味を確定できないような状況に於いて，条文に対する論理式のみでなく，判例や証拠などで示される状況的知識のもとで推論を行う方法であり，司法や法令の運用の場面における適用を目指したもの

である [11][12].

これに対し，本稿で述べるものは，立法の立場から，国民年金法のように基本的には解釈の幅のない法令を対象にして，法令作成者の意図した法令を作るために形式的手法を適用しようとするものである．法令の記述に論理式を使う点では両者は共通であるが，目指す方向は異なる．

本論文はこのような観点から，国民年金法の一部を対象として，法令文の述語論理による形式化と SMT ソルバー Z3Py によるその正しさの検証の可能性について行った試みの報告である．幾つかの基本的な条文を例にして，その論理式化の方法と検証の方法について述べ，今後の課題について考察した．

2　国民年金法の概要

国民年金法は，全 10 章からなる本則と附則からなる．本則には国民年金法の全体が記述されており，附則には本則に伴って必要とされる付随的な規定，例えば，経過規定や施行期日などが規定されている．本論文では，本則のみについて考える．国民年金法の全テキストはオンラインで読むことが可能である [20]．なお，これだけでは必ずしもその内容の理解が困難な部分もあるが，解説書，例えば [21] などを利用することができる．以下は国民年金法の概略である．

第一章　総則（第一条―第六条）

　年金事業の概略の説明，保険料納付済期間，免除期間などの用語の定義

第二章　被保険者（第七条―第十四条の五）

　被保険者の資格，被保険者期間，年金手帳・原簿などの規定

第三章　給付

第一節　通則（第十五条―第二十五条）

　各種の給付（老齢基礎年金，障害基礎年金，遺族基礎年金，付加年金・寡婦年金・死亡一時金）に共通な年金の支給期間及び支払期月，併給の調整，年金の支払の調整などの規定

第二節　老齢基礎年金（第二十六条―第二十九条）

　老齢基礎年金の支給要件，年金額，改定率，支給の繰下げ，失権などの規定（以下，第三，四，五節についても同様）

第三節　障害基礎年金（第三十条―第三十六条の四）

第四節　遺族基礎年金（第三十七条―第四十二条）

第五節　付加年金・寡婦年金・死亡一時金（第四十三条―第六十八条）

第六節　給付の制限（第六十九条―第七十三条）

第四章から第九章（第七十四条―第百十四条）

　年金事業を実施するための制度や運用に関する規定（積立金，費用，不服申し立て，罰則など）

第十章　国民年金基金及び国民年金基金連合会（第百十五条―第百四十八条）

国民年金は，基本的には所定の期間，保険料を納付した人（被保険者）に年金が給付される制度

であるが，もう少し詳しく見ると次のような仕組みになっている．

被保険者であるためにはその資格が必要であり，3 種類の資格と被保険者であった期間が定められている（第二章）．被保険者は，保険料を納付することが義務づけられ，その期間は第一章総則で定義されている．ここでは納付の免除に関することも規定されている．

年金の給付（第三章）には，大きく分けて 4 種類があり，給付の条件などは種類ごとに異なるが，給付の仕方（給付の開始や停止など）に関する共通事項が第一節通則に述べられている．老齢基礎年金については，保険料納付期間と免除期間の合計が 25 年以上と言うのが給付の条件であり，これにより基本的には受給権が発生するが，手続き上は被保険者の請求に基づく厚労大臣裁定により年金の支払いが行われる．受給権の消滅は受給権者の死亡による．なお，受給権が存在する期間でも，他の年金の併給などにより給付の支給が停止される期間もある．この他，第四章以降に，年金事業を実施するための制度や運用などのための諸規則が決められている．

3　述語論理による典型的条文の記述と検証

第一章総則，第二章被保険者，第三章第一節通則，同第二節老齢基礎年金の主要部分を述語論理による論理式化を行ったが，本章では，論理式化のツール，典型的な条文の論理式化と検証の事例について述べる．

3.1　論理式記述体系と SMT ソルバー Z3Py

法令文を具体的に述語論理式として記述するための体系としては，述語論理式の充足性判定ツール Z3 を Python 言語から使うためのインターフェースである Z3Py で使われているものを用いた．Z3Py における論理式の記述の仕方は，以下の法令記述の実例から明らかであるので説明は省略する [15][16]．誤解の無い限り，論理式記述言語，証明ツール，また，それらを併せたものも Z3Py と呼ぶことにする．

述語論理に対する多くの定理証明系があるなかで SMT ソルバー Z3Py を選択した理由は，プログラムのテスト実行感覚で論理式の検査ができ，法務実務者にも受け入れられやすいと考えたこと以外に，述語名に日本語の名前を使うことが可能なことによる．これは日本語で書かれた法文の形式化を行う際は非常に重要であり，筆者の調べた他の定理自動証明系では不可能であった．

Z3Py で用いられている定理証明方式は充足性判定方式 SMT である．述語論理式に対する充足性判定とは，述語 P(x,y,...,z) が真になるような自由変数 x,y,...,z の値が存在するかを判定する方法であり，もし存在すれば充足可能 (satisfiable, sat) といい，そうで無い場合は充足不能 (unsatisfiable, unsat) という．SMT(Satifiability Modulo Theories) は，命題論理式に対する高速な充足性判定技術 SAT と対象問題領域（Z3 の場合は基本的には，ビットベクトル，整数と実数）上の主に線形制約解決アルゴリズムを用いて充足性判定を簡便かつ高速に行う方式である．SAT と SMT の理論や実装，および，Z3/Z3Py の使用法については参考文献を参照されたい [13][14][15][16][17].

3.2　被保険者の資格

3.2.1　条文と論理式化

国民年金法の対象となるのは被保険者であり，3 種類の被保険者が存在することが第七条で定められている．場合分けによる用語の定義のための典型的な条文のひとつとして，その Z3Py による論理式表現を示す．

（被保険者の資格）
第七条　次の各号のいずれかに該当する者は，国民年金の被保険者とする．
一　日本国内に住所を有する二十歳以上六十歳未満の者であつて次号及び第三号のいずれにも該当しないもの（厚生年金保険法（昭和二十九年法律第百十五号）に基づく老齢を支給事由とする年金たる保険給付その他の老齢又は退職を支給事由とする給付であつて政令で定めるもの（以下「厚生年金保険法に基づく老齢給付等」という．）を受けることができる者を除く．以下「第一号被保険者」という．）
二　厚生年金保険の被保険者（以下「第二号被保険者」という．）
三　第二号被保険者の配偶者であつて主として第二号被保険者の収入により生計を維持するもの（第二号被保険者である者を除く．以下「被扶養配偶者」という．）のうち二十歳以上六十歳未満のもの（以下「第三号被保険者」という．）
２　前項第三号の規定の適用上，主として第二号被保険者の収入により生計を維持することの認定に関し必要な事項は，政令で定める．
３　前項の認定については，行政手続法（平成五年法律第八十八号）第三章（第十二条及び第十四条を除く．）の規定は，適用しない．

```
# 第七条被保険者の資格.py
from 第十四条年金原簿 import  *
被保険者=(lambda d:
  Or(第一号被保険者(d),
     第二号被保険者(d),
     第三号被保険者(d)))
第一号被保険者=(lambda d:
  And(二十歳以上六十歳未満(d),
      日本国内に住所を有する(d),
      Not(厚生年金保険法老齢等受給可能(d)),
      Not(第二号被保険者(d)),
      Not(第三号被保険者(d))))
```

```
第二号被保険者=(lambda d:
  厚生年金保険の被保険者(d))
第三号被保険者=(lambda d:
  And(二十歳以上六十歳未満(d),
      被扶養配偶者(d)))
被扶養配偶者=(lambda d:
  And(第二号被保険者の配偶者(d),
     主に第二号被保険者の収入により生計維持(d),
      Not(第二号被保険者(d))))
```

上は，条文を，第二項・第三項を除き，Z3Py における論理式表現に直したものである．正確には，条文で定義される 5 個の用語“被保険者”，“第一号被保険者”，“第二号被保険者”，“第三号被保険者”，“被扶養配偶者”に対応して，日 d を引数とし，それらの内容を表す論理式を値として取る 5 個の関数が lambda 式により定義されている．

その意味は，例えば，“被保険者 (d)”は，対象とする人が日 d において被保険者であることを表す．本論文では，以後，このような関数を述語と呼ぶ．第七条の条文だけからはこれらが日の関数であることは明らかではないが，これらを参照する他の条文，例えば，第十一条（被保険者期間の計算）などからこれを知ることができる．

これらの述語は，対象となる人についてのパラメータを持つことが直感的であるが，国民年金法に関する記述を通して，複数の相異なる人が同時に出現することが希であるので，人に関するパラメータを省略した形をとっている．すなわち，これらの述語は任意に選んだ特定の人についてのものである．

述語定義は，本稿で報告する形式化では，条文ごとにモジュール（ファイル）としてまとめられているが，これらが他のモジュールで必要なときには，そこに import されて参照される．例えば，上の被保険者の資格に関連する述語の定義は，モジュール“第七条被保険者の資格.py”にまとめられているが，そこでは，“二十歳以上六十歳未満”，“日本国内に住所を有する”，“厚生年金保険法老齢等受給可能”，“厚生年金の被保険者”，“第二号被保険者の配偶者”，“主に第二号被保険者の収入により生計維持”の 6 つの述語が参照されているが，それらは“第十四条年金原簿”モジュールで定義されている．そこには，対象としている被保険者に関するデータが記述されている．

3.2.2 被保険者の資格に関する検証

被保険者の資格に関するこの条文は，内容的に分かりにくいところはあまりないが，例えば，当然期待されている性質，「一人の被保険者が同時に異なる種別（号）の被保険者になることはない」は次のようにして確認することができる．

```
# 検証スクリプト 1.py
```

```
from 第七条被保険者の資格 import *
d=Int('d')
種別の重複=(lambda d:
  Or(
   And(第一号被保険者(d),第二号被保険者(d)),
   And(第一号被保険者(d),第三号被保険者(d)),
   And(第二号被保険者(d),第三号被保険者(d))))
s=Solver()
s.add(種別の重複(d))
print(s.check())
# unsat   (上記 print の結果)
```

上の検証スクリプトは，“種別の重複(d)”を充足させる日 d が存在するか否かの決定を，起動した SMT ソルバー s に行わせるものである．その結果は充足不能を表す unsat であり，結局，一人の被保険者が複数の号の被保険者の資格を満たす日 d は存在しないことを表している．

述語“種別の重複”は，最終的に第十四条年金原簿モジュールから import される被保険者に関する現状・履歴データを参照するが，それらは検証スクリプトに関するテストデータの役割を果たしている．充足性判定の結果は，あくまでもこのテストデータのもとでのものであり，その範囲内では種別の重複がなかった事を表している．

ここで用いた下記の年金原簿では，述語はいずれも日を表す整数型変数 d に関する述語であり，例えば，被保険者は日 201 から日 599 まで日本国内に住所を有していた事が表されている．なお，日 d は 26 才の誕生日を起点として数えた日数としている．

```
# 第十四条年金原簿.py
# 第七条被保険者の資格テスト用
from 全体的設定 import *
二十歳以上六十歳未満
    =(lambda d:And(20<=年齢(d),年齢(d)<60))
日本国内に住所を有する
    =(lambda d:And(200<d,d<600))
厚生年金保険法老齢受給可能
    =(lambda d:False)
厚生年金保険の被保険者
    =(lambda d:And(300<d,d<500))
第二号被保険者の配偶者
    =(lambda d:And(400<d,d<550))
```

```
主に第二号被保険者の収入により生計維持
    =(lambda d:And(400<d,d<550))
年齢=(lambda d:26+d/360)
```

3.3　年金の支給期間及び支払期月

3.3.1　条文と論理式化

第十八条は年金の支払や停止に関する規則を述べている．これは日と月という二つの時間軸を含んだ条文であり，このような条文は他にも複数存在する．以下の論理式表現では，記述の簡単化のために“年金給付支給”を単に“支給”，“年金給付支給停止”を“停止”と書く．

（年金の支給期間及び支払期月）

第十八条　年金給付の支給は，これを支給すべき事由が生じた日の属する月の翌月から始め，権利が消滅した日の属する月で終るものとする．

２　年金給付は，その支給を停止すべき事由が生じたときは，その事由が生じた日の属する月の翌月からその事由が消滅した日の属する月までの分の支給を停止する．ただし，これらの日が同じ月に属する場合は，支給を停止しない．

３　年金給付は，毎年二月，四月，六月，八月，十月及び十二月の六期に，それぞれの前月までの分を支払う．ただし，前支払期月に支払うべきであつた年金又は権利が消滅した場合若しくは年金の支給を停止した場合におけるその期の年金は，その支払期月でない月であつても，支払うものとする．

```
# 第十八条年金の支給期間.py
from 第十四条年金原簿 import *
d1=Int('d1')          # 支給事由発生日
d2=Int('d2')          # 受給権利消滅日
d3=Int('d3')          # 停止事由発生日
d4=Int('d4')          # 停止事由消滅日
支給=(lambda m:
    And(受給権月(m),Not(停止(m))))
受給権月=(lambda m:
    Exists(d1,
      And(支給事由発生(d1),属月(d1)+1<=m,
        ForAll(d2,
          Implies(And(受給権利消滅(d2),d1<=d2),
```

```
                    m<=属月(d2))))))

停止=(lambda m:
    Exists(d3,
      And(停止事由発生(d3), 属月(d3)+1<=m,
        ForAll(d4,
          Implies(And(停止事由消滅(d4),d3<=d4),
                  m<=属月(d4))))))
年金支払額=(lambda m:
    If(偶数月(m),
          年金支給額(m-2)+ 年金支給額(m-1),0))
年金支給額=(lambda m:
    If(支給(m), 年金額(m),0))
```

第七条が同一時点での述語の関係（詳細化）を述べたものであるのに対して，第十八条は異なる時点での述語の関係を述べており，ある事由 (要件）の成立の結果として別の時点で他の事由（効果）が成立する形になっている．ここに現れる時点は日と月である．

本来であれば，これらの日と月はカレンダーに基づく実際の日付を使うべきであるが，今回形式化を行った法令文の範囲内では具体の日付に関係する内容が含まれなかったので，日，月ともにある起点からの順序数（整数）によって番号づけられていると単純化した．また，上記述語定義中，関数“属月”は，ある日の属する月を表し，1 月 30 日，1 年 360 日と簡略化し，属月(d)=d/30，としてある．なお，簡単のために，第三項の「ただし」以後で述べられている標準的でない支払期月については省略した．

```
# 全体的設定.py
from z3 import *
属月=(lambda d:d/30)
偶数月=(lambda m:m%2==0)
```

3.3.2　年金の支給期間・支払期月に関する検証

以下では，第十八条の検証について考えるが，そこで参照されている 4 つの述語と 1 つの関数は，年金原簿モジュールで用意する．検証内容として，同じ月に支給と停止が共に真になることがないこと，および，停止の翌月に支給されることがありえることを確認している．

```
# 第十四条年金原簿.py
# 第十八条年金の支給期間テスト用
from 全体的設定 import *
支給事由発生=(lambda d:d==10000)
受給権利消滅=(lambda d:d==14000)
停止事由発生=(lambda d:d==10500)
停止事由消滅=(lambda d:d==12000)
年金額=(lambda m:60000)
```

```
# 検証スクリプト 2.py
from 第十八条年金の支給期間 import *
m1=Int('m1')
m2=Int('m2')
s=Solver()
s.add(支給 (m1))
s.add(停止 (m2))
s.push()
s.add(m1==m2) # 同じ月に支給と停止が成立するか
print(s.check())
# unsat  支給と停止が同時に成立する月はない
s.pop()
s.add(m1==m2+1) # 停止後支給されることがあるか
print(s.check())
print(s.model())
# sat[m2=400,m1=401]
# 月 400 は停止, 月 401(=12000/30+1) は支給
```

3.4 通則と各論

第十八条は，年金の支払や停止に関する一般的な規定であり，通則として述べられている．ここで定義されている 2 つの述語“支給”と“停止”は，第十五条で挙げられている 4 種類の年金（老齢基礎年金，障害基礎年金，遺族基礎年金，付加年金・寡婦年金及び死亡一時金）に共通なものであるが，参照されている 4 つの述語“支給事由発生”，“受給権利消滅”，“停止事由発生”，“停止事由消滅”は，それぞれの年金ごとに個別の定義が与えられている．例えば，老齢基礎年金について

は，第三章第二節においてそれらが“支給要件”などとして規定されており，老齢基礎年金の支給や停止を考える場合には，それらによって“第十八条年金の支給期間.py”中の対応する述語を置き換えなければならない．

これを無理なく行うひとつの方法は，第十八条において，（１）年金の種別に関するパラメータ k を新たに導入し，述語の定義を一般化すると同時に，（２）そこで参照される通則述語と個別の年金で定義される個別述語を，種別パラメータ k を通して結合することである．

具体的には，第十八条で参照している通則述語とそれに対応する個別述語の対応規則を定義する“通則個別述語結合規則”モジュールを作成し，これを第十八条モジュールに import することである．これにより，通則モジュールと個別年金モジュールを独立に構成し，かつ，それらを結合規則モジュールにより関係づけることができる．

概略を示すと次のようである．ちなみに，“第十八条年金の支給期間_通則版”モジュールで参照される述語“支給事由発生”は，“通則個別述語結合規則_第十八条”モジュールにおいて定義され，パラメータ k の値が ’老齢’ の場合は，“老齢基礎年金”モジュール中の“支給要件”が渡される．

```
# 第十八条年金の支給期間_通則版.py
# k:年金種別パラメータ
from 通則個別述語結合_第十八条 import *
...
支給=(lambda k:(lambda m:
   And(受給権月(k)(m),Not(停止(k)(m)))))
受給権月=(lambda k:(lambda m:
   Exists(d1,
    And(支給事由発生(k)(d1),属月(d1)+1<=m,
     ForAll(d2,
      Implies(And(受給権利消滅(k)(d2),d1<=d2),
              m<=属月(d2)))))))
...
```

```
# 通則個別述語結合規則_第十八条.py
import 老齢基礎年金
import 障害基礎年金
...
支給事由発生=(lambda k:
   老齢基礎年金.支給要件 if k=='老齢' else
   障害基礎年金.支給要件 if k=='障害' else
```

...）

...

3.5　年金原簿

第十四条では，国民年金事業の運営に必要な被保険者に関する情報が年金原簿に記録されることを述べており，氏名，資格の取得・喪失，種別の変更，保険料納付状況，基礎年金番号その他厚生労働省令で定める事項を記録するとしている．一方，既に述べたように，本稿における述語論理による記述は任意に選んだ一人の被保険者に関して行っており，したがって，ここでは，年金原簿はこの被保険者に関するものとして与えている．

また，現実の年金業務における情報の管理は別にして，ここでは，年金原簿を介して参照されるものも含めて，条文で参照される被保険者に関する情報は，すべて年金原簿に記録されていると単純化して論理式化を行った．したがって，本稿で提案する法令のテスト・検証は，条文の論理式表現＋年金原簿に対して行われ，年金原簿はテストデータとして扱われる．被保険者の多様な状況に対応して条文の正しさを確認するには，それらを反映した複数の年金原簿データを用意する必要がある．

3.6　第二十条併給の調整の考察

3.6.1　併給の調整メカニズム

年金にはその種別によって併給が可能なものとそうでないものがある．例えば，老齢基礎年金は老齢厚生年金と併給が可能であるが，障害基礎年金とは併給が不可能である．第二十条では，併給不可能な年金の組み合わせと，併給の事由が生じた場合の選択のメカニズムが述べられている．

第二十条　遺族基礎年金又は寡婦年金は，その受給権者が他の年金給付（付加年金を除く.）又は厚生年金保険法 による年金たる保険給付（当該年金給付と同一の支給事由に基づいて支給されるものを除く．以下この条において同じ.）を受けることができるときは，その間，その支給を停止する．老齢基礎年金の受給権者が他の年金給付（付加年金を除く.）又は同法 による年金たる保険給付（遺族厚生年金を除く.）を受けることができる場合における当該老齢基礎年金及び障害基礎年金の受給権者が他の年金給付（付加年金を除く.）を受けることができる場合における当該障害基礎年金についても，同様とする．

２　前項の規定によりその支給を停止するものとされた年金給付の受給権者は，同項の規定にかかわらず，その支給の停止の解除を申請することができる．ただし，その者に係る同項に規定する他の年金給付又は厚生年金保険法 による年金たる保険給付について，この項の本文若しくは次項又は他の法令の規定でこれらに相当するものとして政令で定めるものによりその支給の停止が解除されているときは，この限りでない．

３　第一項の規定によりその支給を停止するものとされた年金給付について，その支給を停止すべき事由が生じた日の属する月分の支給が行われる場合は，その事由が生じたときにおいて，当該年金給付に係る前項の申請があつたものとみなす．
４　第二項の申請（前項の規定により第二項の申請があつたものとみなされた場合における当該申請を含む．）は，いつでも，将来に向かつて撤回することができる．

年金法の標準的解釈書である文献 [21] によると，第一項の条文中“受けることができる”は“受給権を取得している”であるとし，次のような解釈を与えている．

- 第一項によると，年金給付 A（例えば，老齢基礎年金）の受給権者が，年金給付 B（例えば，障害基礎年金）の受給権も取得すると，A の支給は停止される．一方，併給不可能な年金の組み合わせは対称的であるので，A の受給権を取得していることにより B も停止され，結局 A も B も支給が停止される．
- 第一項により，いったんすべての年金給付が支給停止となった場合，第二項により，受給権者は自分の希望する年金給付について支給停止の解除を申請することにより，それを選択する．
- 第三項は，第一項の規定により，支給停止となる年金給付について，既に支給されている年金給付がある場合には，申請がない限り，引き続きその年金を支給するためのものである．

3.6.2　第三項の問題点

併給に関する上記規則の論理式化の過程で，第三項には問題があることが判明した．以下にそれを述べ，3.6.3 では Z3Py による確認を行う．

A の受給権を既に取得し年金の支給を受けているときに，月 m に属する日 d において新たに B の受給権を取得したとする．このとき，第一項により月 m+1 は A と B ともに支給停止状態になるが，B を選択するために日 d に B の停止解除申請を行うと，B は月 m+1 は支給される．

一方，A は月 m において支給されているので，第三項により日 d に停止解除申請が行われたものとみなされ，月 m+1 は A も支給されることになる．結局，月 m+1 は A も B もともに支給されてしまうことになる．

この問題は，もちろん，第三項がなければ起こらない．もし A の支給を継続する場合は，（B の停止解除申請を行わずに）A の停止解除申請のみを行えばよい．

なお，現実の運用としては，B が受給可能になったときに，B の受給請求の代わりに“年金受給選択申出書”[22] を提出し，その中で A か B かの選択を行うようになっており，問題は起こらない．上記問題点の指摘は，あくまでも条文通りに法令を適用した場合に起きる問題である．

3.6.3　定理証明器による確認

ここでは，上記問題点の指摘を定理証明器により確認する．日 10000 において受給権が発生した年金 A の受給中に，A と併給不可能な年金 B の受給権が日 10500 に発生し，さらに同日に B の停

止解除申請がなされた場合，検証結果は，翌月の月 351(=10500/30+1) において年金 A と B がともに支給されることを示している．（なお，第三項が無ければ月 351 において，B のみが支給されることも同様に検証される．）

```
# 第十八条年金の支給期間_年金 AB.py
from 全体的設定 import *
from 通則個別述語結合規則_年金 AB import *
d1=Int('d1')              # 支給事由発生日
d2=Int('d2')              # 受給権利消滅日
d3=Int('d3')              # 停止事由発生日
d4=Int('d4')              # 停止事由消滅日
支給=(lambda k:(lambda m:
  And(受給権月 (k)(m),Not(停止 (k)(m)))))
受給権月=(lambda k:(lambda m:
  Exists(d1,
    And(支給事由発生 (k)(d1), 属月 (d1)+1<=m,
     ForAll(d2,
      Implies(And(受給権利消滅 (k)(d2),d1<=d2),
              m<=属月 (d2)))))))
停止=(lambda k:(lambda m:
  Exists(d3,
    And(停止事由発生 (k)(d3), 属月 (d3)+1<=m,
     ForAll(d4,
      Implies(And(停止事由消滅 (k)(d4),d3<=d4),
                  m<=属月 (d4)))))))
```

```
# 通則個別述語結合規則_年金 AB.py
from 年金原簿_年金 AB import *
another=(lambda k:'B' if k=='A' else 'A')
支給事由発生=受給権発生
停止事由発生=(lambda k:
   受給権発生 (another(k)))
停止事由消滅=(lambda k:(lambda d:
   Or(停止解除申請 (k)(d),            # 第二項
      停止解除みなし申請 (k)(d))))    # 第三項
```

```
停止解除みなし申請=(lambda k:(lambda d:
   And(k==’A’, 支給事由発生 (’B’)(d))))
                                # 補題 AB により
```

```
# 年金原簿_年金 AB.py
from z3 import *
受給権発生=(lambda k:(lambda d:
   Or(And(k==’A’,d==10000),
      And(k==’B’,d==10500))))
受給権利消滅=(lambda k:(lambda d:
   Or(And(k==’A’,d==14000),
      And(k==’B’,d==12000))))
停止解除申請=(lambda k:(lambda d:
   Or(And(k==’A’,d==12000),
      And(k==’B’,d==10500))))
```

```
# 検証スクリプト３.py
from 第十八条年金の支給期間_年金 AB import *
d,m=Ints(’d m’)
r=Solver()
# 補題 AB
for k in [’A’,’B’]:
  r.add(ForAll(d,
    Implies(And(k==’A’, 支給事由発生 (’B’)(d)),
            And(停止事由発生 (k)(d),
                支給 (k)(属月 (d))))))
print(r.check())  # sat, 補題 AB 成立
s=Solver()
s.add(And(支給 (’A’)(m), 支給 (’B’)(m)))
print(s.check())  # sat [m=351]
# 月 351 は年金 A,B ともに支給
```

上記検証スクリプト 3 においては，まず補題 AB の証明を行い，その結果を利用して併給不可能性の検証を行っている．この理由は，今回の述語定義方式では，再帰的な述語の定義が不可能な事

によるものである (4.4.1 参照)．すなわち，“通則個別述語結合規則_年金 AB.py”において，

```
停止解除みなし申請=(lambda k:(lambda d:
    And(停止事由発生 (k)(d),
        支給 (k)(属月 (d)))))
```

とするのが条文の文章上は直接的であるが，この場合，述語“支給”の定義が再帰的になる．これを避けるために，この例題では年金 A の受給権が B より早く確立していることを考慮し，補題 AB を経由して，“通則個別述語結合規則_年金 AB.py”のようにしている．

4 考察

4.1 論理式化から見た国民年金法

国民年金法全 165 条（第十章国民年金基金及び国民年金基金連合会，附則を除く）を論理式記述の観点からその内容を調べると，

1. 保険料の納付や給付などの年金に関する規則が書かれており論理式化が有効なもの
2. 被保険者，厚労大臣，年金機構，自治体などの間の関係が書かれたもの
3. 年金財政，罰則などの年金政策や他の法令との関連が書かれたもの

が，それぞれ約 7 : 4 : 5 の割合で存在することが分かる．このうち，今回，論理式記述を行ったのは，上記 1. のうちで国民年金法の基本的部分である，被保険者の定義，保険料免除，給付，停止，併給などを扱う五条，七条，十一条，十四条，十八条，二十条，二十六条，八十九条，九十条，九十条の二，九十条の三，および，老齢基礎年金に関する第二十六条～二十九条の計 15 条である．

このように，本稿で対象にした論理式記述は限定されたものではあるが，法令文の正しさの確保に論理式記述と定理自動証明技術が有効であることの基本的確認や，このような技術が現実に用いられるための幾つかの知見が得られた．

4.2 法令論理式化の意義

国民年金法などの法令は，行政手続きとして運用され，それに準拠する情報システムが正しく構築されなければならないことから，法令は正確かつ明確に記述されなければならない．この点において，自然言語による記述には限界があり，述語論理などによる形式的記述には大きな利点がある．

法令の形式的記述の最大の利点は，記述の正しさや満たすべき性質のチェックを，定理証明器などの推論ツールを使って機械的に行うことが可能なことであり，これは，法令文に含まれる誤りや問題点の除去などに非常に有効である．本稿においても，第二十条の条文に誤りがあることを指摘し，定理証明器によりその確認を行うことができることを示した．条文の論理的深度は深くなく，

このような問題点の指摘を行う上で，現在の自動定理証明ツールは十分な能力があると考えられる．[*1]

法令論理式の検証は，テストデータ（国民年金法の場合は年金原簿）と検証スクリプトとともに定理自動証明システム上で行われるが，その具体的な設計にはプログラムやソフトウェアのテストや検証に関してこれまでに蓄積された膨大な技術が利用可能である．法令の論理式化は，このような技術を正しい法令の作成に利用することを可能とするものである．

4.3 論理式記述の作成

4.3.1 法令文と論理式記述

国民年金法のような法令文は，一般の文書に比べると明確な記述が心懸けられている．しかしながら，法令文は，対象領域に関する知識や常識をもつ法務実務者が読むことを前提としたものであり，推論エンジンが機械的に解釈する論理式記述との間には差があることに留意する必要がある．

例えば，第二十条では，ある年金の支給停止が，他の年金を“受けることができる”ときに起きることを述べている．このとき，“受けることができる”の定義がなされておらず，本稿では文献 [21] との整合性が取れるように，“受給権が確立している”と解釈した．そして，これを，給付の停止を定義している第十八条第二項の“停止すべき事由が生したとき”と対応付けた．同様に，同条の“停止事由の消滅”を第二十条第二項・第三項で述べられている停止解除申請と対応付けて論理式記述を行った．これらの対応付は条文には陽には書かれておらず，論理式記述を行うにはこれらを推定する必要がある．

4.3.2 基本的記述方策

基本述語の設定

論理式記述では基本となる述語の設定がまず必要であるが，国民年金法については，その細部は別にすると，基本的な事柄は法令自体にかなり整理されて書かれており，条文ごとにその内容を理解することにより論理式記述を行う事ができる．

年金問題では，概略的には，被保険者の資格の成立，受給条件の成立，年金の給付という順にイベントが発生し，これらに関連するほとんどの述語は日付（日または月）のパラメータを持つことになる．これを理解すると基本的な述語の設定をおこなうことが出来る．ただし，3.2.1 でも触れたように，条文によっては必ずしもそれが陽には書かれていないこともあり，イベントの流れを理解し関連条文との整合性が取れるように適切なパラメータの設定を行う必要がある．

単一人原則

今回の記述では，やはり 3.2.1 で述べたように，述語から人に関するパラメータを省略するという原則を取り，述語定義を簡単化している．これが可能であったのは，複数の人が本質的に関与するような条文が無いことによるものである．

[*1] 筆者が利用した Z3Py システム (バージョン 4.4.1(最新バージョンは 4.8.0)，2.6GHz Intel Core i5) では，本稿にある検証スプリクトは瞬時に実行が終了する．

複数の人が軽度に関わる条文はいつか存在するが，そのいずれもが，関与するの人の年金原簿間の整合性や参照という問題に帰着させることにより，この原則を守った記述が可能であった．例えば，第七条では，ある人 A が第三号被保険者であるためには，別の第二号被保険者 B の配偶者であることが要求されるが，これは A,B の年金原簿が婚姻関係を正しく反映していることに帰着される．また，保険料の免除申請に関する第九十条，第九十条の二では，申請の条件として本人以外にも世帯主や配偶者の経済的困窮を条件としているが，これは，世帯主や配偶者の年金原簿の参照により解決される．

論理式構成

今回論理式記述を行った法令文に記述されている内容の多くは，(1) 場合分けによる用語の定義と，(2) ある条件の成立を待って新たな条件が成立することの記述である．前者に関しては，論理演算子による論理式の構成が，また，後者に関しては条件が成立する日付を特定するために，日付上の束縛変数に関する量限定子 Exists,ForAll が使われている．

4.3.3 パターン化

各条文はそれぞれ異なる内容について述べているが，その表現については似ている点も多く，それらは一つのパターンとして論理式記述することにより理解しやすく，また，構造的な記述が可能になる．

第十八条では，受給権の存在する期間と支給停止期間では，細部は異なっているが，その構造は以下の“期間”ような形になっている．

```
期間=(lambda f,g:(lambda m:
  Exists(d1,
     And(f(d1),属月(d1)+1<=m,
         ForAll(d2,
            Implies(And(g(d2),d1<=d2),
                    m<=属月(d2)))))))
```

ここで，f,g は期間の初日 d1 と最終日 d2 を定める述語である．これを全体的設定モジュールに定義しておくと，第十八条の主要部分は次のように簡潔に記述可能である．

```
支給=(lambda m:
     And(受給権月(m),Not(停止(m))))
受給権月=期間(支給事由発生,受給権利消滅)
停止=期間(停止事由発生,停止事由消滅)
```

今回論理式化を行ったなかでは，上記以外にも日付に関係していくつかのパターンが見つかっている．

4.4 述語論理記述体系 Z3Py

4.4.1 日本語名述語

日本の法令を記述するためには，日本語の名前を使って述語定義を行えることは必須であり，これが今回 Z3Py を採用した大きな理由のひとつである．Z3Py は定理証明系 Z3 を Python 言語から使うためのインターフェースであり，この点に関して 2 つの機能がある．一つは，Python プログラムの実行により論理式を構成し変数の値として設定できること，もう一つは，このように構成した論理式を定理証明系 Z3 に引き渡し，その充足可能性判定を行わせることである．

このとき，Python3 言語では変数名に日本語を使うことができるため，論理式を値とする関数式（lambda 式）を日本語名変数にバインドすることにより，日本語名の述語定義とすることができる．Python は関数展開を行い本体のみを Z3 証明器に引き渡すので，最終的に展開された関数本体が Z3 の受け付ける ASCII 文字からなる論理式になっていれば，Z3 証明系により評価可能となる．なお，この述語定義の方法は，通常の述語論理におけるそれより制限されたものであり，再帰的に述語を定義することは出来ない．

4.4.2 高階関数

今回の論理式化では，論理体系としては一階述語論理の範囲内にとどまったが，親言語である Python のもつ高階関数定義機能の利用が，法令文を構造的に記述するうえで極めて有効であることを確認した．例えば，高階関数により定義された日付オペレータ“期間”により，期間に関する記述を簡潔に行うことが可能となること，また，“第十八条年金の支払期間_通則”にあるように，高階関数が通則の自然な記述を可能にすることが確認された．

4.4.3 モジュール化

多数の条文からなる法令の論理式表現において，論理式集合をフラットな構造として管理することは，その証明や変更などを行う上で大きな問題である．法令の持つ構造 (章，節，条など）にしたがって述語定義を構造化する上で，Python 言語の提供するモジュール化の機能は有効である．なお，今回の試行では条文ごとのテストを行ったので条文ごとにモジュール化を行ったが，もう少し大きなまとまりを一つのモジュールに設定することが適当であろう．

今回は行わなかったが，年金システム全体という立場からは政府や自治体，年金機構などの組織を含めた形式化が必要である．事実，4.1 で見たように，全条文の三分の一はこれらの関係についての規則が書かれている．このような規則の形式的記述と検証には，例えば，オブジェクト間の制約記述のための体系などが有効であろう [19]．

4.5 SMT ソルバー

従来の法令の論理式記述の研究は，司法や法令の運用局面に関するものがほとんどであり，条文に含まれる論理の積み重ねにより結論を導く過程を，形式論理における定理導出過程として捉えるものが主である．これが SMT のような決定アルゴリズムを使って解を求めるソルバーが，これらの研究で利用されなかった理由であろうと考える．

このソルバーは上のような目的には向いていないが，推論規則や公理系などの知識を必要とせずに，実データに基づきテスト感覚で法令の検証を高速に行うことが出来るので，立法の過程で法令作成者が使う道具としては適している考えられる．さらに，一度論理式記述が完成すれば，ソルバーを法令シミュレータとして利用することも可能であろう．

本稿で述べた検証事例は，いずれも論理式を充足させる整数上の変数値の決定という形で行われ，年金原簿には具体的数値によってテストデータが与えられていた．一方，Z3Py には関数変数や抽象的に定義したデータ領域上の変数を対象にした充足性判定を行う機能があり [17]，これを利用するとより一般的な検証が可能な場合がある．例えば，検証スクリプト 1 で用いた年金原簿において，

日本国内に住所を有する
```
=(lambda d:And(200<d,d<600))
```

などのテスト述語に代えて

日本国内に住所を有する
```
=Function('f',IntSort(),BoolSort())
```

のように“日本国内に住所を有する”などを関数変数として充足性判定を行うことにより，一般的な条件のもとでの充足不可能性の結論を得ることができる．

4.6 論理式記述のスケールアップ

4.6.1 立法者のための論理式作成環境

本稿で提唱する方法を現実に適用するには，立法者自身が論理式の作成を行えることが必要であろう．述語論理自身は比較的直感的な概念であるが，複雑な条文を含む法令の論理式記述には，その構造的記述を容易にするパターンやモジュール化機構などの記述体系が必要である，これらの体系やそれを支えるツールがあれば，彼ら自身が現実の法令の論理式記述を行えるのではないかと期待される．

ツールとしては，条文を作るごとにその論理式化と検証を行えるような形でアジャイルに法令の作成とテストを可能とするものが望ましい．場合によっては，逆に論理式を先に作り，対応する条文はそれから自動生成することも考えられる．また，大規模な法令を対象とするには複数の法務実

務者が協調して論理式記述を行う必要があり，そのための作業環境（レポジトリや版管理機能など）が必須であるが，これらは現在ソフトウェア開発で用いられているものを利用できると考える．

4.6.2　自然言語処理による論理式化

機械的処理により既存の法令の論理式表現を得る，あるいは，そのための前処理を行えることは社会的な意義が大きい．島津は，国民年金法の条文を例として，イベント変数を用いる Davidsonian style により単語ごとに論理式を割り当てる方式により，条文を述語論理に変換する方法の検討を行っている [8]．同様な方法で，一般の自然言語文を述語論理式に翻訳するシステムについての方法が田中らにより提案されている [23]．

条文とその論理式表現の間にはギャップがあることは既に指摘したが，用語の選択や文章表現に関する統一性や正確性が高いという条文の特徴を利用し，このような手法により正しい論理式を自然言語処理技術により得ることは今後の課題であろう．

5　おわりに

述語論理による国民年金法の記述と定理自動証明器による検証の可能性に関するケーススタディについて述べた．いくつかの基本的な条文を例に取り，条文の論理式による記述方法，SMT ソルバー Z3Py による条文の正しさの確認方法，論理式記述の方法論やスケールアップなどについて述べた．また，年金の併給に関する条文に誤りがあることの確認を Z3Py の適用により行うことにより，このような方法が誤りの無い法令を作成する上で有効であることを具体的に示した．

なお，今回は国民年金法をとりあげたが，ここで得られた知見は国民年金法に特有なものではなく，法律の大多数を占める行政的な手続きに関する法令に適用できるものであると考えられる．

謝辞 本論文に関して熱心な議論をしていただいた中央大学研究開発機構法令工学ユニットおよび JAIST 大学院大学調査研究機構法令工学研究グループの皆様に感謝いたします．特に，島津明北陸先端科学技術大学院大学名誉教授には論文全体の構成と自然言語処理に関して，養老真一大阪大学法学部教授には条文の解釈に関して多くをご教示頂きました．また，査読者には適切なご意見を多数頂いたことに感謝いたします．なお，本研究は公益財団法人セコム科学技術振興財団一般研究助成（法令工学に基づく法令作成・検証の基盤構築，角田篤泰）からの援助を受けています．

参考文献

[1] 職業安定局文書: http://www.mhlw.go.jp/file/05-Shingikai-10201000-Daijinkanbousoumuka-Soumuka /0000046988.pdf

[2] 榎並利博: 近年の“立法爆発”で法律は“スパゲティ状態”の限界に, 法と経済のジャーナル, http://judiciary.asahi.com/fukabori/2015091500002.html

[3] 片山卓也（編）:法令工学の提案,JAIST Press(2007)

[4] 片山卓也:法律の作成にソフトウェア開発技術を,p35, 情報処理 Vol.58 No.1,2017
[5] 角田篤泰: ソフトウェア工学との類似性に着目した立法支援方法 (1), 名古屋大學法政論集, 235 号, pp.41-99, 2010 年 (http://ir.nul.nagoya-u.ac.jp/jspui/handle/2237/14329).
[6] 角田篤泰, 島亜紀, 齋藤大地, 大谷忠:全国自治体例規データベース eLen の構築と定量的例規調査, 情報ネットワーク・ローレビュー, 第 13 巻, 第 1 号,pp.14-33(2014)
[7] 島津明:国民年金法の構造的書き換えー法令工学の立場からー,JAIST Press(2009)
[8] 島津明:法令工学の提案, 第 2 章法令文書の言語処理,pp.21-50,JAIST Press(2007)
[9] Sergot,M.J., Sadri,F., Kowalski,R.A., Kriwaczek,F., Hammond,P., Cory,H.T., "The British Nationality Act as a Logic Program", Communications of the ACM,Vol.29 No.5,pp.370-386(1986)
[10] 新田克己，長尾順太郎，水鳥哲也:工業所有権法の知識表現システム KRIP，情報処理学会論文誌 27 巻 11 号,pp.1042-1052(1986)
[11] Satoh, K., Asai, K., Kogawa, T., Kubota, M., Nakamura, M., Nishigai, Y., Shirakawa, K., Takano, C.: "PROLEG: An Implementation of the Presupposed Ultimate Fact Theory of Japanese Civil Code by PROLOG Technology", New Frontiers in Artificial Intelligence: JSAI-isAI 2010 Workshops, Revised Selected Papers, LNAI 6797, pp. 153-164 (2012)
[12] 東条敏,Wong,S., 新田克己，横田一正:状況論理による法的推論の形式化, 情報処理学会論文誌 36 巻 1 号,pp51-60(2007)
[13] 酒井政裕, 今井健男:SAT 問題と他の制約問題との相互発展, コンピュータソフトウェア, Vol.32, No.1, pp.103-119(2015)
[14] Moura,L.:SMT Solvers: Theory and Implementation,Summer School on Logic and Theorem Proving, https://leodemoura.github.io/files/oregon08.pdf
[15] Moura,L.:Z3 guide:http://rise4fun.com/Z3/tutorial /guide
[16] Moura,L.:Z3Py guide:Z3 API in Python,http: //cpl0.net/ argp/papers/z3py-guide.pdf
[17] Moura,L.:Z3 Advanced Topics, http://www.cs.tau.ac.il /msagiv/courses/asv/z3py /advanced-examples.htm
[18] Moura,L.,Bjorner,N.:Proofs and Refutations, and Z3, https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/nbjorner-iwil08.pdf
[19] OCL:Object constraint language 2.0(2006) http://www,omg.org/spec/OCL/2.2/PDF/
[20] 国民年金法, http://www.law.e-gov.go.jp /htmldata/S34/S34HO141.html
[21] 国民年金法逐条解釈（平成 27 年 4 月 1 日現在施行法令準拠）, 厚生労働省 http://www.mhlw.go.jp/file/06-Seisakujouhou-12500000- Nenkinkyoku /0000095039.pdf
[22] http://www.nenkin.go.jp/service/jukyu/tetsuduki /kyotsu/20150501.html

[23] 田中リベカ, 峯島宏次,Pascual Martinez-Gomez, 宮尾祐介, 戸次大介:日本語 CCG パーザに基づく意味解析・推論システムの提案, 言語処理学会 第 22 回年次大会 発表論文集,pp757-760(2016)

第 II 部

実践編
国民年金法基本条文の論理式記述と検証の実践

6　論理式の表記と検証方法

6.1　論理式の表記

条文をどのようにして論理式化するかの基本的方法については，第 I 編 4.3.2 に述べたが，ここでは，条文に対する論理式を具体的に表記する方法について，第七条 (7.2 参照) を例に取り説明する．

- 第七条では 4 つの基本的な用語（被保険者，第一号被保険者，第二号被保険者，第三号被保険者）と一つの補助的な用語（被扶養配偶者）が規定されている．論理式記述ではそれらを述語論理における述語として定式化する．これらの述語は，日から論理値 `True`,`False` への関数であり，例えば，「**第一号被保険者 (d)**」は，もし対象とする人が日 `d` において第一号被保険者であれば `True` となり，そうでなければ `False` となる．
- 第一号被保険者は第一項によりその内容が規定されているが，それを定式化したものが述語**第一号被保険者**の定義式である．

```
第一号被保険者=(lambda d:
        And(二十歳以上六十歳未満 (d),
            日本国内に住所を有する (d),
            Not(厚生年金保険法老齢等受給可能 (d)),
            Not(第二号被保険者 (d)),
            Not(第三号被保険者 (d))))
```

これは，任意の日 `d` に対して，**第一号被保険者 (d)** が `True` になるのは，**二十歳以上六十歳未満 (d)**，**日本国内に住所を有する (d)** がいずれも `True` であり，かつ，**厚生年金保険法老齢等受給可能 (d)**, **第二号被保険者 (d)**, **第三号被保険者 (d)** がいずれも `False` の場合であることを述べたものである．

- 日は，ある開始日を起点とした日数によって表現しており，`d` は整数型の変数である．変数としては，日には `d,d1,d2,...` などを, また，月については，`m,m1,m2,...` などを用いている．
- 上の述語定義において，`lambda` 式の本体は Z3Py の論理式であり，他の述語や論理演算子から構成される．`And`，`Not` は Z3Py の論理演算子であり，Python の論理演算子 `and`，`not` とは異なる．論理演算子としてはこのほか `Or,Imply,==,ForAll,Exists` などを用いることが出来る．

- 参照されている述語**第二号被保険者，第三号被保険者**も第七条で同様に定義されるが，これら第七条で定義される述語は一つのファイル f7.py（Python モジュール) に格納されている．一方，**二十歳以上六十歳未満，日本国内に住所を有する，厚生年金保険法老齢等受給可能**は，対象とする被保険者の状態を記述した述語であり，年金原簿モジュール (例えば，honnin_v7.py) に記録されている．
- f7 モジュールでは，上記論理式の定義式の他に，Z3 モジュール，年金原簿モジュールをインポートする必要がある．また，一般には，定義に使われる基本述語や関数を集めた基本用語定義集モジュール definitions もインポートされる．これらは essentials モジュールおよび検証スクリプト v7.py において指定され，

  ```
  from essentials import *
  ```

  の実行により一括してインポートすることが出来る．詳細については，「検証スクリプトにおける年金原簿，基本用語定義集の設定」(12) を参照されたい．

6.2 検証方法

第七条「被保険者の資格」の論理式記述 f7 を例に取り（7.2.2 参照)，検証方法を述べる．
検証は検証スクリプトを実行することにより行われる．f7 では日 d に関する 5 個の述語が定義されているが，そのうち以下の 3 つの述語

第一号被保険者 (d)，第二号被保険者 (d)，第三号被保険者 (d)

は，いかなる日 d においても高々ただ一つしか成立しないことが期待される．以下はこれを検証するための検証スクリプト v7 である（7.2.3 参照)．

```
# v7.py

# 以下，honnin を指定
import dummy
dummy.honnin='honnin_v7'
from essentials import *  # z3,honnin_v7 をインポート
from f7 import *
d=Int('d')
s=Solver()
p=Or(And(第一号被保険者 (d), 第二号被保険者 (d)),
     And(第一号被保険者 (d), 第三号被保険者 (d)),
     And(第二号被保険者 (d), 第三号被保険者 (d)))
s.add(p)
```

```
print(s.check())
# unsat
```

- s には起動されたソルバー割り当てられ，s.add(p) はそれに検査したい論理式 p を与える．s.check() は，p を True にする d が存在するか否かを検査するものであり，存在すれば sat(satisfiable, 充足可能) が，存在しなければ unsat(unsatisfiable, 充足不能) が返される．この場合は unsat である．
- ここで使われた年金原簿モジュールは honnin_v7 であり，これは essentials で z3 と共にインポートされる．基本用語定義マクロは使用されないので，インポートする必要が無い．
- d は Z3 の使用する変数であり，Int は整数型の変数を作り出すための関数である．ForAll,Exists の束縛変数をはじめ充足性判定の対象となる論理式に含まれる変数には，自由変数，束縛変数を問わずこのような型定義文が必要である．ちなみに，最終的に検証される論理式は lambda を含むことは出来ず，lambda 式の仮引数は検証前に実引数により具体化されていなければならない．
- 検証は，検証スクリプトモジュール v7 をコマンド

  ```
  python3 v7.py
  ```

  により実行することにより行われる．
- なお，検証に使われる Z3Py システムの最新版は，以下のコマンドの実行によって取得できる．

  ```
  python3 -m pip install z3-solver
  ```

7　条文と論理式

7.1　第五条　用語の定義

7.1.1　条文

この法律において、「保険料納付済期間」とは、第七条第一項第一号に規定する被保険者としての被保険者期間のうち納付された保険料（第九十六条の規定により徴収された保険料を含み、第九十条の二第一項から第三項までの規定によりその一部の額につき納付することを要しないものとされた保険料につきその残余の額が納付又は徴収されたものを除く。以下同じ。）に係るもの、第七条第一項第二号に規定する被保険者としての被保険者期間及び同項第三号に規定する被保険者としての被保険者期間を合算した期間をいう。

２　この法律において、「保険料免除期間」とは、保険料全額免除期間、保険料四分の三免除期間、保険料半額免除期間及び保険料四分の一免除期間を合算した期間をいう。

３　この法律において、「保険料全額免除期間」とは、第七条第一項第一号に規定する被保険者としての被保険者期間であつて第八十九条第一項、第九十条第一項又は第九十条の三第一項の規定により納付することを要しないものとされた保険料に係るもののうち、第九十四条第四項の規定により納付されたものとみなされる保険料に係る被保険者期間を除いたものを合算した期間をいう。

４　この法律において、「保険料四分の三免除期間」とは、第七条第一項第一号に規定する被保険者としての被保険者期間であつて第九十条の二第一項の規定によりその四分の三の額につき納付することを要しないものとされた保険料（納付することを要しないものとされた四分の三の額以外の四分の一の額につき納付されたものに限る。）に係るもののうち、第九十四条第四項の規定により納付されたものとみなされる保険料に係る被保険者期間を除いたものを合算した期間をいう。

５　この法律において、「保険料半額免除期間」とは、第七条第一項第一号に規定する被保険者としての被保険者期間であつて第九十条の二第二項の規定によりその半額につき納付することを要しないものとされた保険料（納付することを要しないものとされた半額以外の半額につき納付されたものに限る。）に係るもののうち、第九十四条第四項の規定により納付されたものとみなされる保険料に係る被保険者期間を除いたものを合算した期間をいう。

６　この法律において、「保険料四分の一免除期間」とは、第七条第一項第一号に規定する被保険者としての被保険者期間であつて第九十条の二第三項の規定によりその四分の一の額につき納付することを要しないものとされた保険料（納付することを要しないものとされた四分の一の額以外の四分の三の額につき納付されたものに限る。）に係るもののうち、第九十四条第四項の規定により納付されたものとみなされる保険料に係る被保険者期間を除いたものを合算した期間をいう。

（７，８，９　省略）

7.1.2 論理式

```
# f5.py　第五条　用語の定義

from essentials import *
from f11 import 第一号被保険者期間, 第二号被保険者期間, 第三号被保険者期間
from f89 import 保険料法定免除要件
from f90 import 保険料全額免除要件
from f90_2 import 保険料四分の三免除要件, 保険料半額免除要件, 保険料四分の一免除要件
from f90_3 import 保険料免除_学生要件

保険料納付済期間=(lambda m:
            Or(And(第一号被保険者期間(m), 保険料納付済(m)),
               第二号被保険者期間(m),
               第三号被保険者期間(m)))
保険料免除期間=(lambda m:
            Or(保険料全額免除期間(m),
               保険料四分の三免除期間(m),
               保険料半額免除期間(m),
               保険料四分の一免除期間(m)))
保険料全額免除期間=(lambda m:
            And(第一号被保険者期間(m),
                Or(保険料法定免除要件(m),
                   保険料全額免除要件(m),
                   保険料免除_学生要件(m)),
                Not(追納により納付とみなされる期間(m))))
保険料全額免除_学生を除く期間=(lambda m:
            And(第一号被保険者期間(m),
                Or(保険料法定免除要件(m),
                   保険料全額免除要件(m)),
                Not(追納により納付とみなされる期間(m))))
保険料四分の三免除期間=(lambda m:
            And(第一号被保険者期間(m),
                保険料四分の三免除要件(m),
                残りの四分の一納付済(m),
                Not(追納により納付とみなされる期間(m))))
保険料半額免除期間=(lambda m:
```

```
            And(第一号被保険者期間(m),
                残りの半額納付済(m),
                保険料半額免除要件(m),
                Not(追納により納付とみなされる期間(m))))
保険料四分の一免除期間=(lambda m:
            And(第一号被保険者期間(m),
                保険料四分の一免除要件(m),
                残りの四分の三納付済(m),
                Not(追納により納付とみなされる期間(m))))
```

7.1.3 検証

```
# v5.py
# 年金原簿 honnin_v5,haiguusha_v5,setainushi_v5 のもとでの第五条の検証

import dummy
dummy.honnin='honnin_v5'
dummy.haiguusha='haiguusha_v5'
dummy.setainushi='setainushi_v5'
dummy.definitions='definitions_simple'

from f5 import *
import time
start=time.time()
m=20
L=充足リスト(保険料納付済期間,m)
print('保険料納付済期間',L)
L=充足リスト(保険料免除期間,m)
print('保険料免除期間',L)
L=充足リスト(保険料全額免除期間,m)
print('保険料全額免除期間',L)
L=充足リスト(保険料四分の三免除期間,m)
print('保険料四分の三免除期間',L)
L=充足リスト(保険料半額免除期間,m)
print('保険料半額免除期間',L)
L=充足リスト(保険料四分の一免除期間,m)
print('保険料四分の一免除期間',L)
```

```
print(' 実行時間',time.time()-start, 'sec')

# 保険料納付済期間 [6, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19]
# 保険料免除期間 [7, 8, 18]
# 保険料全額免除期間 [18]
# 保険料四分の三免除期間 []
# 保険料半額免除期間 [7, 8]
# 保険料四分の一免除期間 []
# 実行時間 2.1401238441467285 sec
```

7.1.4 ノート

- 第五条では，保険料納付済期間や免除期間が定義されている．これらは，納付や免除が行われた月の集合であり，第二十六条や二十七条において老齢基礎年金の支給要件や年金額を算定する場合の基礎になるものである．これらの期間の定義には，
  - 被保険者期間を決定する規則（第十一条）
  - 各種の保険料免除要件を定める規則（第八十九、九十条、九十条の二，九十条の三）
  - 保険料納付に関する記録 (年金原簿)

  が必要になる．

  なお，保険料の納付については，正常な納付の他に，滞納者に対する督促を経て徴収されたもの（第九十六条），申請により免除された保険料が後で納付さたもの（追納，第九十四条）が含まれること，また，一部免除の場合の残りの保険料の納付は保険料納付済期間には含まれないことが述べられている．
- 条文の論理式化 `f5` においては，条文モジュール `f11,f89,f90,f90_2,f90_3` から必要な述語がインポートされる．さらに年金原簿がインポートされ，述語 **保険料納付済，追納により納付とみなされる期間，残りの四分の一・半額・四分の三納付済**が参照されている．なお，年金原簿のインポートは，

  ```
  from essentials import *
  ```

  において行われている．(12 参照)
- 論理式化は条文の内容をそのまま論理式に直したものではあるが，字義的な変換を行う事は困難で条文の内容の理解が必要である．例えば，条文では第七条について言及されているが，必要なのは第十一条でありこれには触れられていない．
- 検証では小さな年金原簿モジュール `honnin_v5`（11 参照）を用い，定義されている納付・免除期間が正しく計算されることを確認した．検証スクリプト中，**充足リスト (f,m)** は述語 `f(i)` が充足される（`True` となる）`i` を最大 `m` 個みつけてリストとして返す関数であり，`definitions_simple` モジュールで定義されている．(10 参照)

7.2　第七条　被保険者の資格

7.2.1　条文

次の各号のいずれかに該当する者は、国民年金の被保険者とする。

一　日本国内に住所を有する二十歳以上六十歳未満の者であつて次号及び第三号のいずれにも該当しないもの（厚生年金保険法（昭和二十九年法律第百十五号）に基づく老齢を支給事由とする年金たる保険給付その他の老齢又は退職を支給事由とする給付であつて政令で定めるもの（以下「厚生年金保険法に基づく老齢給付等」という。）を受けることができる者を除く。以下「第一号被保険者」という。）

二　厚生年金保険の被保険者（以下「第二号被保険者」という。）

三　第二号被保険者の配偶者であつて主として第二号被保険者の収入により生計を維持するもの（第二号被保険者である者を除く。以下「被扶養配偶者」という。）のうち二十歳以上六十歳未満のもの（以下「第三号被保険者」という。）

２　前項第三号の規定の適用上、主として第二号被保険者の収入により生計を維持することの認定に関し必要な事項は、政令で定める。

３　前項の認定については、行政手続法（平成五年法律第八十八号）第三章（第十二条及び第十四条を除く。）の規定は、適用しない。

7.2.2 論理式

```
# f7　第七条　被保険者の資格

from essentials import *
# 以下は，honnin モジュールから import される.
# 日本国内に住所を有する，二十歳以上六十歳未満，厚生年金保険法老齢等受給可能，
# 厚生年金保険の被保険者，主に第二号被保険者の収入により生計維持

被保険者=(lambda d:
            Or(第一号被保険者 (d),
               第二号被保険者 (d),
               第三号被保険者 (d)))
第一号被保険者=(lambda d:
            And(日本国内に住所を有する (d),
                二十歳以上六十歳未満 (d),
                Not(第二号被保険者 (d)),
                Not(第三号被保険者 (d)),
                Not(厚生年金保険法老齢等受給可能 (d))))
第二号被保険者=(lambda d:
            厚生年金保険の被保険者 (d))
第三号被保険者=(lambda d:
            And(被扶養配偶者 (d),
                二十歳以上六十歳未満 (d)))
被扶養配偶者=(lambda d:
            And(第二号被保険者の配偶者 (d),
                主に第二号被保険者の収入により生計維持 (d),
                Not(第二号被保険者 (d))))
```

7.2.3 検証

```
# v7.py
# 第七条の検証
# 年金原簿 honnin_v7 のもとでの検証
import dummy
dummy.honnin='honnin_v7'
from essentials import *
from f7 import *
d=Int('d')

# 日 450 は，第何号被保険者か？
s=Solver()
s.add(第一号被保険者 (450))
print(s.check())
# unsat

s=Solver()
s.add(第二号被保険者 (450))
print(s.check())
# sat

s=Solver()
s.add(第三号被保険者 (450))
print(s.check())
# unsat

def 被保険者の種類 (d):
    s1=Solver()
    s2=Solver()
    s3=Solver()
    s1.add(第一号被保険者 (d))
    s2.add(第二号被保険者 (d))
    s3.add(第三号被保険者 (d))
    if s1.check()==sat: return '第一号被保険者'
    elif s2.check()==sat: return '第二号被保険者'
    elif s3.check()==sat: return '第三号被保険者'
```

```
print(被保険者の資格 (450))  # 第二号被保険者

# 日 450 以外に第二号被保険者になりえるか？
t=Solver()
t.add(第二号被保険者 (d),d!=450)
if t.check()==sat:
    print(t.model())
# [d=368]

# 同時に異なる種別の被験者になり得るか？
s=Solver()
p=Or(And(第一号被保険者 (d), 第二号被保険者 (d)),
     And(第一号被保険者 (d), 第三号被保険者 (d)),
     And(第二号被保険者 (d), 第三号被保険者 (d)))
s.add(p)
print(s.check())
# unsat

# honnin_v7 においては，第一号，第二号，第三号被保険者である日は，それぞれ
# 第一号被保険者：日 201〜日 300 および 日 550〜日 599
# 第二号被保険者：日 301〜日 499
# 第三号被保険者：日 500〜日 549
# であるが，これは以下のようであるが，これは次のようにして確認できる．

t=Solver()
p1 = ForAll(d, 第一号被保険者 (d)==Or(And(201<=d,d<=300),And(550<=d,d<=599)))
p2 = ForAll(d, 第二号被保険者 (d)==And(301<=d,d<=499))
p3 = ForAll(d, 第三号被保険者 (d)==And(500<=d,d<=549))
t.add(p1,p2,p3)
print(t.check())
# sat
```

```
################
# honnin_v7.py #
################

# この年金原簿は，第七条，第八条，第九条，第十一条の検証で使用される．
# 日 d は，26 才の誕生日を起点とする通日で表現．1 年 360 日，1 月 30 日と簡単化
# なお，「年齢」の扱いに関しては，一般的なものが definition_date_3 に与えられている．

from essentials import *

年齢=(lambda d:26+d/360)
二十歳以上六十歳未満=(lambda d:And(20<=年齢(d),年齢(d)<60))
日本国内に住所を有する=(lambda d:And(200<d,d<600))
厚生年金保険法老齢等受給可能=(lambda d:False)
厚生年金保険の被保険者=(lambda d:And(300<d,d<500))
第二号被保険者の配偶者=(lambda d:And(400<d,d<550))
主に第二号被保険者の収入により生計維持=(lambda d:And(500<=d,d<550))

#    *----日本在住-------------------------------------------------*
#    |                                                            |
#    |            *-----企業に勤務---------------*                |
#    |            |                               |               |
#    |            |              *----結婚-----------------*      |
#    |            |              |                |          |    |
# ---*------------*--------------*----------------*----------*----*------
# d=201          301            401              499        549      599
#    |            |                               |          |        |
#    *--第一号----**----------第二号-------------**-第三号---**-第一号--*
# m= 6           9 10                      15      16   17     18    19

# なお，以下は honnin_v7 を第十一条期間の検証で用いた場合の各被保険者期間を示す．

# 第一号被保険者期間 [6，7，8，9，18，19]
# 第二号被保険者期間 [10，11，12，13，14，15]
# 第三号被保険者期間 [16，17]
```

```
# v7_func
# 第七条の検証
# 同時に異なる種別の被保険者にはなれないことを，v7 の場合のように具体的な
# 年金原簿に対してではなく，より一般的に検証する．
# "日本国内に住所を有する”などを述語変数として扱い，これににいかなる述語を代入
# しても充足出来ないことを示す．

import dummy
dummy.honnin='honnin_v7_func'
from essentials import *
from f7 import *
d=Int('d')
s=Solver()
p=Or(And(第一号被保険者 (d), 第二号被保険者 (d)),
     And(第一号被保険者 (d), 第三号被保険者 (d)),
     And(第二号被保険者 (d), 第三号被保険者 (d)))
s.add(p)
print(s.check())
# unsat
```

```
# honnin_v7_func.py
# v7_func で使用
from essentials import *

二十歳以上六十歳未満=Function('p',IntSort(),BoolSort())
日本国内に住所を有する=Function('f',IntSort(),BoolSort())
厚生年金保険法老齢等受給可能=Function('q',IntSort(),BoolSort())
厚生年金保険の被保険者=Function('g',IntSort(),BoolSort())
第二号被保険者の配偶者=Function('h',IntSort(),BoolSort())
主に第二号被保険者の収入により生計維持=Function('k',IntSort(),BoolSort())
```

7.2.4 ノート

- 条文の内容を忠実に論理式に直したものである．なお，条文中の２および３については，これらを考慮して「主として第二号被保険者の収入により生計を維持すること」の認定が適切

の行われ，年金原簿に反映されていると仮定した．

- 検証は 3 種類の年金原簿データに関して行った．
  1. honnin_v7
     具体的な日データとしの年金原簿．日本国内に住所を有するなどは，具体的な述語としてあたえられている．
  2. honnin_v7_func
     関数変数を許す充足性判定を利用した検証のための年金原簿．日本国内に住所を有するなどが関数 IntSort→BoolSort を表す変数として定義されており，v7_func ではこれらの変数への述語の割り当てに関する充足性判定により，honnin_v7 より一般的な検証が行われている．
  3. honnin_v7_abs
     抽象日付けデータ型を利用した検証のための年金原簿．具体的な内容に関しては，8.2 において述べる．
- 法令改正により第七条は第 1 項三号が本稿掲載のものから変更されている．

7.3　第八条　資格取得の時期

7.3.1　条文

前条の規定による被保険者は、同条第一項第二号及び第三号のいずれにも該当しない者については第一号から第三号までのいずれかに該当するに至つた日に、二十歳未満の者又は六十歳以上の者については第四号に該当するに至つた日に、その他の者については同号又は第五号のいずれかに該当するに至つた日に、それぞれ被保険者の資格を取得する。

一　二十歳に達したとき。

二　日本国内に住所を有するに至つたとき。

三　厚生年金保険法に基づく老齢給付等を受けることができる者でなくなつたとき。

四　厚生年金保険の被保険者の資格を取得したとき。

五　被扶養配偶者となつたとき。

7.3.2　論理式

```
# f8.py　第八条　資格取得の時期
# 以下は，条文の構造に従って内容を忠実に論理式化したものである.
# AND,OR,NOT, 成立に至る：definitions において定義

from essentials import *
from f7 import 第二号被保険者, 第三号被保険者, 被扶養配偶者

被保険者資格取得=(lambda d:
            If(NOT(OR(第二号被保険者, 第三号被保険者))(d),
                  OR(第一号, 第二号, 第三号)(d),
                  If(二十歳未満の者又は六十歳以上の者 (d),
                        第四号 (d),
                        OR(第四号, 第五号)(d))))

第一号=成立に至る (lambda d:年齢 (d)==20)
第二号=成立に至る (日本国内に住所を有する)
第三号=成立に至る (NOT(厚生年金保険法老齢等受給可能))
第四号=成立に至る (厚生年金保険の被保険者)
第五号=成立に至る (被扶養配偶者)
二十歳未満の者又は六十歳以上の者=(lambda d:Or(年齢 (d)<20,60<=年齢 (d)))
```

7.3.3 検証

```
# v8.py
# 年金原簿 honnin_v7 をテストデータとする検証

import dummy
dummy.honnin='honnin_v7'
dummy.definitions='definitions_simple'

from essentials import *
from f7 import 被保険者
from f8 import 被保険者資格取得

d=Int('d')
s=Solver()
p=ForAll(d,Implies(成立に至る(被保険者)(d), 被保険者資格取得(d)))
s.add(p)
print(s.check()) # sat
t=Solver()
q=ForAll(d,Implies(被保険者資格取得(d), 成立に至る(被保険者)(d)))
t.add(q)
print(t.check()) # unsat

# 実際，成立に至る(被保険者)(d) は，d=201 に対してのみ True となるが，
# 被保険者資格取得(d) は，d=201,301,500 に対して True となる.
u=Solver()
u.add(成立に至る(被保険者)(d))
print(u.check()) # sat
print(u.model()) # d=201
u.add(d!=201)
print(u.check()) # unsat

v=Solver()
v.add(被保険者資格取得(d))
print(v.check()) # sat
print(v.model()) # d=301
v.add(d!=301)
```

```
print(v.check()) # sat
print(v.model()) # d=500
v.add(d!=500,d==201)
print(v.check()) # sat
print(v.model()) # d=201
```

7.3.4 ノート

- この条文は，（前日まで成立していなかった）被保険者の資格がある日 d に成立に至るための条件についてのべたものである．すなわち，論理式で書けば，

  And(Not(被保険者 (d-1)), 被保険者 (d))

  あるいは，マクロ表現

  成立に至る=lambda f:lambda d:And(Not(f(d-1),f(d)))

  を使うと（10 参照），

  成立に至る (被保険者)(d)

  である．したがって，この条文は新しい内容を付加するものではなく，第七条から機械的に判断できることを明文化したものである．
- f8 は，条文の構造に従ってその内容を論理式に変換したものである．条文のタイトルから，任意の日 d に関して，

  被保険者資格取得 (d) ⇔ 成立に至る (被保険者)(d)

  を期待したが，検証の結果これは成立しない．

  成立に至る (被保険者)(d) ⇒ 被保険者資格取得 (d)

  は成立するが，逆の

  被保険者資格取得 (d) ⇒ 成立に至る (被保険者)(d)

  は成立しない．被保険者資格取得 (d) が実際に被保険者の資格を取得する日より多くの日にちを規定していることは具体の検証結果からも分かる．
- 第七条によれば第一号被保険者の資格を取得するには，年齢，住所，厚生年金受給に関する 3 つの条件が同時に成立することが必要であるが，本条文ではこれらの 3 条件の「いずれかに該当するに至った日に，資格が取得される」としている．この表現は誤解を招きやすい．「資格が取得されるのは，いずれかに該当するに至った日に限る」とすれば明確である．

7.4　第九条　資格喪失の時期

7.4.1　条文

第七条の規定による被保険者は、次の各号のいずれかに該当するに至つた日の翌日（第二号に該当するに至つた日に更に第七条第一項第二号若しくは第三号に該当するに至つたとき又は第三号から第五号までのいずれかに該当するに至つたときは、その日）に、被保険者の資格を喪失する。

一　死亡したとき。

二　日本国内に住所を有しなくなつたとき（第七条第一項第二号又は第三号に該当するときを除く。）。

三　六十歳に達したとき（第七条第一項第二号に該当するときを除く。）。

四　厚生年金保険法に基づく老齢給付等を受けることができる者となつたとき（第七条第一項第二号又は第三号に該当するときを除く。）。

五　厚生年金保険の被保険者の資格を喪失したとき（第七条第一項各号のいずれかに該当するときを除く。）。

六　被扶養配偶者でなくなつたとき（第七条第一項第一号又は第二号に該当するときを除く。）。

7.4.2　論理式

```
# f9.py　第九条　資格喪失の時期

from essentials import *
from f7 import 被扶養配偶者, 第一号被保険者, 第二号被保険者, 第三号被保険者, 被保険者

被保険者資格喪失=(lambda d:
        If(OR(AND(第二号,OR(第二号被保険者, 第三号被保険者)),
              OR(第三号, 第四号, 第五号))(d),
           True,
           OR(第一号, 第二号, 第三号, 第四号, 第五号, 第六号)(d-1)))

六十歳=lambda d:本人.年齢(d)==60
死亡=lambda d:False
第一号=成立に至る(死亡)
第二号=成立に至る(AND(NOT(日本国内に住所を有する),
                       NOT(OR(第二号被保険者, 第三号被保険者))))
第三号=成立に至る(AND(六十歳,NOT(第二号被保険者)))
第四号=成立に至る(AND(厚生年金保険法老齢等受給可能,
                       NOT(OR(第二号被保険者, 第三号被保険者))))
```

第五号=成立に至る (AND(NOT(厚生年金保険の被保険者),NOT(被保険者)))
第六号=成立に至る (AND(NOT(被扶養配偶者),
NOT(OR(第一号被保険者, 第二号被保険者))))

7.4.3 検証

```
# v9.py
# 年金原簿 honnin_v7 のもとでの検証

import dummy
dummy.honnin='honnin_v7'
dummy.definitions='definitions_simple'

from essentials import *
from f7 import 被保険者
from f9 import 被保険者資格喪失
d=Int('d')

t1=Solver()
t2=Solver()
p1=ForAll(d,Implies(成立に至る (NOT(被保険者))(d), 被保険者資格喪失 (d)))
p2=ForAll(d,Implies(被保険者資格喪失 (d), 成立に至る (NOT(被保険者))(d)))
t1.add(p1)
t2.add(p2)
print(t1.check()) # sat
print(t2.check()) # unsat
```

7.4.4 ノート

- この条文は,（前日成立していた）被保険者の資格がある日 d に成立しなくなることのための条件についてのべたものである. これは論理式として記述すると,

  And(被保険者 (d-1),Not(被保険者 (d)))

  あるいは,

  成立に至る (NOT(被保険者))(d)

  である.
- 条文は論理式 被保険者資格喪失 (d) を定義しているが, これと上の 成立に至る (NOT(被保険者))(d) との関係については, 以下が成立することが検証結果から分かる.

  成立に至る (NOT(被保険者))(d) ⇒ 被保険者資格喪失 (d)

は成立するが，逆の

被保険者資格喪失 (d) ⇒ 成立に至る (NOT(被保険者))(d)

は成立しない．

すなわち，被保険者資格喪失は 成立に至る (NOT(被保険者)) のための必要条件であるが，必要十分条件ではない．

- 条文の文章表現に関しては次のように解釈した．
  - 本文中「A に至った日の翌日（B のときは，その日)」は，B が成立したときはその日，B が成立せず A に至ったときはその翌日，と解釈した．
  - 本文で参照している第二号から第六号は，いずれも，「P に至るとき（Q である場合を除く)」と言う表現であるが，これは，

    成立に至る (AND(P,NOT(Q)))

    と解釈した．一見 AND(成立に至る (P),NOT(Q)) のようにも読めるが，この時は被保険者資格喪失が 成立に至る (NOT(被保険者)) の必要条件にならないことが検証されてしまう．
- 「日本国内に住所を有する」などの被保険者の要件は，死亡していないことを内容的に前提としているが，第七条ではそれが陽には述べられていない．したがって，第一号の死亡のケースについては，成立に至る (NOT(被保険者)) との関係を形式的に検証することは意味がないと考え，死亡 (d)=False として検証を行った．

7.5　第十一条，第十一条の二　被保険者期間の計算

7.5.1　条文

　第十一条　被保険者期間を計算する場合には、月によるものとし、被保険者の資格を取得した日の属する月からその資格を喪失した日の属する月の前月までをこれに算入する。

２　被保険者がその資格を取得した日の属する月にその資格を喪失したときは、その月を一箇月として被保険者期間に算入する。ただし、その月にさらに被保険者の資格を取得したときは、この限りでない。

３　被保険者の資格を喪失した後、さらにその資格を取得した者については、前後の被保険者期間を合算する。

　第十一条の二　第一号被保険者としての被保険者期間、第二号被保険者としての被保険者期間又は第三号被保険者としての被保険者期間を計算する場合には、被保険者の種別（第一号被保険者、第二号被保険者又は第三号被保険者のいずれであるかの区別をいう。以下同じ。）に変更があつた月は、変更後の種別の被保険者であつた月とみなす。同一の月において、二回以上にわたり被保険者の種別に変更があつたときは、その月は最後の種別の被保険者であつた月とみなす。

7.5.2　論理式

```
# f11　第十一条，第十一条の二　被保険者期間の計算

from essentials import *
from f7 import 被保険者, 第一号被保険者, 第二号被保険者, 第三号被保険者

被保険者資格取得=(lambda d:
        And(Not(被保険者 (d-1)), 被保険者 (d)))
被保険者資格喪失=(lambda d:
        And(被保険者 (d-1),Not(被保険者 (d))))
被保険者期間=期間 2(被保険者資格取得, 被保険者資格喪失)
第一号被保険者期間=(lambda m:
        And(被保険者期間 (m),
            月内で最後に成立 (第一号被保険者, 第二号被保険者, 第三号被保険者)(m)))
第二号被保険者期間=(lambda m:
        And(被保険者期間 (m),
            月内で最後に成立 (第二号被保険者, 第一号被保険者, 第三号被保険者)(m)))
第三号被保険者期間=(lambda m:
        And(被保険者期間 (m),
            月内で最後に成立 (第三号被保険者, 第一号被保険者, 第二号被保険者)(m)))
```

7.5.3 検証

```
# v11 第十一条の検証
# 年金原簿 honnin_v7 のもとでの検証
import dummy
dummy.honnin='honnin_v7'
dummy.definitions='definitions_simple'

from f11 import *
import time
start=time.time()
d=Int('d')
solve(被保険者資格取得(d)) # d=200
solve(被保険者資格喪失(d)) # d=601
L1=充足リスト(第一号被保険者期間,30)
L2=充足リスト(第二号被保険者期間,30)
L3=充足リスト(第三号被保険者期間,30)
L=充足リスト(被保険者期間,30)
print("被保険者期間",L)
# [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
print("第一号被保険者期間",L1)
# [6, 7, 8, 9, 18, 19]
print("第二号被保険者期間",L2)
# [10, 11, 12, 13, 14, 15]
print("第三号被保険者期間",L3)
# [16, 17]

# ある月が同時に複数の異なる被保険者期間に属することはないことの検証
m=Int('m')
s=Solver()
s.add(Or(And(第一号被保険者期間(m),第二号被保険者期間(m)),
         And(第一号被保険者期間(m),第三号被保険者期間(m)),
         And(第二号被保険者期間(m),第三号被保険者期間(m))))
print(s.check())
# unsat
print(time.time()-start,'sec')
# 1.5498101711273193 sec
```

```
# v11_ladyA_date
# 年金原簿　honnin_ladyA_date のもとでの検証
import dummy
dummy.honnin='honnin_ladyA_date'
dummy.definitions='definitions_date_3'
from essentials import *
from f11 import * # 第十一条被保険者期間の計算
import time
m=Int('m')

start=time.time()
L1=充足リスト(第一号被保険者期間,500)
L1=[mjm2ym(mjm) for mjm in L1]
print("第一号被保険者期間",'\n',L1)
print(len(L1)) #134
print(time.time()-start,'sec','\n')
# 0.460216760635376 sec

start=time.time()
L2=充足リスト(第二号被保険者期間,500)
L2=[mjm2ym(mjm) for mjm in L2]
print("第二号被保険者期間",'\n',L2)
print(len(L2)) #318
print(time.time()-start,'sec','\n')
# 0.6563467979431152 sec

start=time.time()
L3=充足リスト(第三号被保険者期間,500)
L3=[mjm2ym(mjm) for mjm in L3]
print("第三号被保険者期間",'\n',L3)
print(len(L3)) #28
print(time.time()-start,'sec','\n')
# 0.2897191047668457 sec

# ある月が同時に複数の異なる被保険者期間に属することはないことの検証
start=time.time()
m=Int('m')
```

```
s=Solver()
s.add(Or(And(第一号被保険者期間(m),第二号被保険者期間(m)),
         And(第一号被保険者期間(m),第三号被保険者期間(m)),
         And(第二号被保険者期間(m),第三号被保険者期間(m))))
print(s.check(),'\n',' 異なる被保険者期間に属する月は無い')
print(time.time()-start,'sec')
# unsat
# 1.5195868015289307 sec
```

7.5.4 ノート

- 被保険者の資格やその種別は日ごとにその成立が決まるが，ここではそれを月ごとの成立として定義する規則を述べている．保険料の納付や免除の記録，年金額の計算の元となる．月に関する要件の成立はそれが成立する日の属する月とするのが基本である．
- definitons において定義されている関数 **期間２，月内で最後に成立** の使用により条文の論理式は大幅に簡略化されている．
- **期間 2(f,g)** は，条件 f を最初に満たす日 d1 の属月から、条件 g を最初に満たす日 d2 の属月の前月まで月の集合を表す．f,g が同じ月に成立するときには、その月は含まれる．もし，このような f,g の対が複数ある場合は、それらの集合を合算したものを表す．
- **月内で最後に成立 (f,g,h)(m)** は，月 m の中では，条件 f,g,h の内で f が最後に成立することを表す．
- 検証 v11 は，年金原簿 honnin_v7 と定義集 definitions_simple のもとで各種期間の定義が意図した通りであること確認すためのものである．v11_ladyA_date は，大きな年金原簿 honnin_ladyA_date と西暦による日付に対応した基本用語定義集 definitions_date_3 の元での検証のためのスクリプトとである．

7.6　第十四条　年金原簿

7.6.1　条文

厚生労働大臣は、国民年金原簿を備え、これに被保険者の氏名、資格の取得及び喪失、種別の変更、保険料の納付状況、基礎年金番号（政府管掌年金事業（政府が管掌する国民年金事業及び厚生年金保険事業をいう。）の運営に関する事務その他当該事業に関連する事務であつて厚生労働省令で定めるものを遂行するために用いる記号及び番号であつて厚生労働省令で定めるものをいう。）その他厚生労働省令で定める事項を記録するものとする。

7.6.2　ノート

- 条文では，国民年金事業の運営に必要な被保険者に関する情報が年金原簿に記録されることを述べており，氏名，資格の取得・喪失，種別の変更，保険料納付状況，基礎年金番号その他厚生労働省令で定める事項を記録するとしている．一方，年金業務の実行には上記以外にも，被保険者の戸籍や税などに関する情報をが必要であり，本稿では，条文で参照される被保険者に関する情報は，すべて年金原簿に記録されていると単純化して論理式化を行った．
- 本稿における述語論理による記述は，任意に選んだ一人の被保険者に関して行っている．したがって，条文の中で年金原簿の情報が参照される場合は，年金原簿は基本的にはこの被保険者に関するものであり，標記上は「本人」という統一的な名前で参照している．保険料免除に関する条文では，本人以外にも配偶者や世帯主の年金原簿が必要になるが，これらはそれぞれ「配偶者」「世帯主」として参照される．
- 年金原簿は，条文論理式のなかで参照される述語定義の集合として与えられる．一例として，第七条の検証で用いた年金原簿 honnin_v7 を以下に示す．

```
年齢=(lambda d:26+d/360)
二十歳以上六十歳未満=(lambda d:And(20<=年齢(d),年齢(d)<60))
日本国内に住所を有する=(lambda d:And(200<d,d<600))
厚生年金保険法老齢等受給可能=(lambda d:False)
厚生年金保険の被保険者=(lambda d:And(300<d,d<500))
第二号被保険者の配偶者=(lambda d:And(400<d,d<550))
主に第二号被保険者の収入により生計維持=(lambda d:And(400<d,d<550))
```

- 本稿での年金原簿は，対象とする条文の検証やテストを行うしたためのものであり，その条文で必要とされる情報のみを記録している．したがって，例えば，保険料の納付や年金の受給に関して年金原簿に記録される述語は月の関数である．保険料の納付済みは，保険料が納付された月のみを記録しており，実際に納付された日は記録していない．
- 条文のテストや検証は，条文論理式＋年金原簿に対する定理証明として行われ年金原簿はテストデータである．

$$\text{条文論理式}+\text{年金原簿} \Rightarrow \boxed{\text{Z3Py 定理証明器}} \Rightarrow \text{検証結果}$$

- 多くの条文は年金原簿に対する問合せや情報の抽出に関するものであり（例えば，資格の有無の判定，年金額の決定など），上のよう形の推論としてその内容を捉えることができる．

  一方，保険料の支払いや免除などに関する条文では，年金原簿に対する操作（保険料納付の記録や免除決定の記録など）が含まれる．年金原簿に対する操作は，その操作により年金原簿がどのように変更されるかを示すことにより規定されるが，具体的には，操作前の年金原簿が満たすべき条件と操作の前後で年金原簿がどのように変化をするかを表す述語を使って定義される．これら 2 つの述語を用いた推論により条文の検証を上と同じ方式で行う事が出来る．

$$\text{条文論理式}\left\{\begin{array}{c}\text{問合せに関する論理式}\\ + \\ \text{操作に関する論理式}\end{array}\right\}+\text{年金原簿} \Rightarrow \boxed{\text{Z3Py 定理証明器}} \Rightarrow \text{検証結果}$$

  具体例については，例えば，追納に関して 7.18 を参照されたい．
- 検証する条文や検証内容によって複数の年金原簿を用いた．
  - honnin_v5：f5 検証ための小さなもの，honnin_v7+ 保険料納付免除
  - honnin_v7：被保険者の資格に関する小さなもの，f7,f8,f9,f11 の検証
  - honin_v7_abs,honnin_v7_func：抽象データ，関数データのもとでの f7 の検証
  - honnin_v18：支給停止事由発生消滅についての小さなもの，f18 の検証
  - honnin_v18_abs：f18 の抽象データのもとでの検証
  - honnin_AB：併給の調整に関する小さなもの，f20 の検証
  - honnin_ladyA_date：A 女史の年金に関する履歴，配偶者 haiguusha_ladyA_date, 世帯主 setainushi_ladyA_dte と共に f26,f27(支給要件，年金額) の検証に基本用語定義 definitions_date と共に利用
  - honnin_v28：支給の繰下 f28 の検証

7.7　第十五条　給付の種類

7.7.1　条文

この法律による給付（以下単に「給付」という。）は、次のとおりとする。

一　老齢基礎年金

二　障害基礎年金

三　遺族基礎年金

四　付加年金、寡婦年金及び死亡一時金

7.7.2　論理式

```
# f15.py　第十五条　給付の種類

給付,(老齢基礎年金, 障害基礎年金, 遺族基礎年金, 付加年金等)
                =EnumSort('kyufu',('rourei','shougai','izoku','huka'))
```

7.7.3　ノート

- 第十五条では４種類の年金を給付の対象として規定し，第十六条〜第二十五条はこれらの年金給付全てが満たすべき一般的規則（通則）を述べている．したがって，これらの通則の論理式記述が自然な形で行えるように第五条「給付の種類」は形式化しなければならない．本稿で行った方法は，（１）給付の種類を表す給付型の定式化，（２）給付型による述語のパラメータ化，（３）通則個別述語結合規則の導入である．
- f15 では，4 つの要素，老齢基礎年金, 障害基礎年金, 遺族基礎年金, 付加年金等からなる数え上げ型給付を定義している．なお，kyufu,rourei,shougai,izoku,huka は，型名やこれら要素名に対する Z3 における表現であり，Z3 はこれらを用いて計算を行い実行結果を出力する．
- 給付型は，第十八条などの通則の論理式記述をパラメータ化し，各給付型値ごとに特殊化された論理式記述を得るために利用される．

7.8　第十八条　年金の支給期間及び支払期月

7.8.1　条文

　年金給付の支給は、これを支給すべき事由が生じた日の属する月の翌月から始め、権利が消滅した日の属する月で終るものとする。

２　年金給付は、その支給を停止すべき事由が生じたときは、その事由が生じた日の属する月の翌月からその事由が消滅した日の属する月までの分の支給を停止する。ただし、これらの日が同じ月に属する場合は、支給を停止しない。

３　年金給付は、毎年二月、四月、六月、八月、十月及び十二月の六期に、それぞれの前月までの分を支払う。ただし、前支払期月に支払うべきであつた年金又は権利が消滅した場合若しくは年金の支給を停止した場合におけるその期の年金は、その支払期月でない月であつても、支払うものとする。

7.8.2　論理式

```
# f18.py　第十八条　年金の支給期間及び支払期月，通則版
from essentials import *

受給権月=期間（支給事由発生，受給権利消滅）
支給=(lambda m:And(Not(停止（m)），受給権月（m)))
停止=期間（停止事由発生，停止事由消滅）
年金支払額=(lambda m:
            If(偶数月（m)，年金支給額（m-2)+ 年金支給額（m-1),0))
年金支給額=(lambda m:If(支給（m)，年金額（m),0))
```

```
# f18_specific.py　第十八条　年金の支給期間及び支払期月，個別版
# k:給付型パラメータ
from essentials import *
from specialization_a18 import * # 通則個別述語対応規則_第十八条

受給権月=(lambda k:期間（支給事由発生（k)，受給権利消滅（k)))
支給=(lambda k:(lambda m:And(受給権月（k)(m),Not(停止（k)(m)))))
停止=(lambda k:期間（停止事由発生（k)，停止事由消滅（k)))
年金支払額=(lambda k:(lambda m:
            If(偶数月（m)，年金支給額（k)(m-2)+ 年金支給額（k)(m-1),0)))
年金支給額=(lambda k:(lambda m:If(支給（k)(m)，年金額（k)(m),0)))
```

7.8.3 検証

```
# v18.py

import dummy
dummy.definitions='definitions_simple'
dummy.honnin='honnin_v18'

from essentials import *
from f18 import * #第十八条年金の支給期間及び支払期月
m1=Int('m1')
m2=Int('m2')
s=Solver()
t=Solver()
r=Solver()
t.add(停止 (10500/30+1))
print('10500/30+1',' 停止',t.check())
# 10500/30+1 停止 unsat
# 停止せず、月 10500/30 には停止事由発生と消滅が共に起きるから。

r.add(停止 (12500/30+1))
print('12500/30+1',' 停止',r.check())
# 12500/30+1 停止 sat

s.add(支給 (m1))
s.add(停止 (m2))
s.push()
s.add(m1==m2)
print(' 支給と停止が同じ月に起きるか？',s.check())
# unsat

s.pop()
s.add(m1==m2+1)
print(' 停止の翌月に支給されることがあるか？',s.check())
print (s.model())
# 停止の翌月に支給されることがあるか？
# sat
```

```
# [m2 = 433,
#  m1 = 434,
#  d1!10 = 12500,
#  d1!9 = 10000,
#  d2!8 = [else -> 13000]]

a=Int('a')
r.add(年金支払額(440)==a)
print('月440の支払額は？',r.check())
print('a=',r.model()[a])

# sat
# a=140000
```

```
# honnin_v18.py
# 第十八条の検証で使用

from essentials import *

支給事由発生=(lambda d:d==10000)
受給権利消滅=(lambda d:d==14000)
停止事由発生=(lambda d:Or(d==10500,d==12500))
停止事由消滅=(lambda d:Or(d==10520,d==13000))
年金額=(lambda m:70000)
```

7.8.4 ノート

- この条文は通則として述べらており，すべての年金種別に対して共通である．論理式 f18 は年金種別パラメータ k を持たず，年金給付一般に関連する検証はこれによって行う．
- 一方，論理式 f18_specific では，年金種別を表す給付型のパラメータ k が導入されている．支給事由発生(k) などの条件は，通則個別述語結合規則_第十八条において，年金種別 k ごとに設定される．パラメータ k は，f15 で定義された数え上げ型給付に属する値を取る．

  支給事由発生=lambda k:(lambda m:年金 k の支給事由発生(m))
- 期間は definitions で定義されているが，期間(f,g) は条件 f を最初に満たす日 d1 の属月の翌月から、条件 g を最初に満たす日 d2 の属月までの月 m の集合を表す．このような f,g の対が複数ある場合は、それらの集合を合算したものである．

- ３の ただし 以後の随時支払い時期については省略されている．逐条解釈 (TAC) によると，この随時支払には次の２つの場合がある．（１）前回の標準支払以降に起きた権利の消滅あるいは支払い停止により，次回の標準支払時に払うべき年金の支払，（２）請求認定の遅れによる支払遅延分の支払．いずれも，年金原簿に記録されていれば論理式記述に於いても対処可能である．
- v18 は，通則版の f18 単独での検証を実データに関して行ったものである．複数の年金が関係する個別版 f18_specific の検証の具体例については，第二十条併給の調整に関連して行った v20 を参照されたい．また，f18 に関して抽象日付けデータを使った記号的検証も行った．（参照 8.2）

7.9　第二十条　併給の調整

7.9.1　条文

遺族基礎年金又は寡婦年金は、その受給権者が他の年金給付（付加年金を除く。）又は厚生年金保険法による年金たる保険給付（当該年金給付と同一の支給事由に基づいて支給されるものを除く。以下この条において同じ。）を受けることができるときは、その間、その支給を停止する。老齢基礎年金の受給権者が他の年金給付（付加年金を除く。）又は同法による年金たる保険給付（遺族厚生年金を除く。）を受けることができる場合における当該老齢基礎年金及び障害基礎年金の受給権者が他の年金給付（付加年金を除く。）を受けることができる場合における当該障害基礎年金についても、同様とする。

２　前項の規定によりその支給を停止するものとされた年金給付の受給権者は、同項の規定にかかわらず、その支給の停止の解除を申請することができる。ただし、その者に係る同項に規定する他の年金給付又は厚生年金保険法による年金たる保険給付について、この項の本文若しくは次項又は他の法令の規定でこれらに相当するものとして政令で定めるものによりその支給の停止が解除されているときは、この限りでない。

３　第一項の規定によりその支給を停止するものとされた年金給付について、その支給を停止すべき事由が生じた日の属する月分の支給が行われる場合は、その事由が生じたときにおいて、当該年金給付に係る前項の申請があつたものとみなす。

４　第二項の申請（前項の規定により第二項の申請があつたものとみなされた場合における当該申請を含む。）は、いつでも、将来に向かつて撤回することができる。

7.9.2　論理式

```
# 第二十条の条文自身は論理式化の対象ではない.
```

7.9.3　検証

```
# v20.py
# 年金 A,B（年金原簿 honnin_AB）を対象とした第二十条「併給の調整」に関する検証
import dummy
dummy.definitions='definitions_simple'

from f18_specific import * #第十八条年金の支給期間，通則版
d,m=Ints('d m')

# 年金 A,B が同じ月 m に支給されることがあることの検証
s=Solver()
```

```
s.add(And(支給 (年金 A)(m), 支給 (年金 B)(m)),m==351)
print(s.check())  #sat
print(s.model())  #[m=351]
#print(充足リスト (AND(支給 (年金 A), 支給 (年金 B)),100))
# 月 351 は年金 A,B ともに支給される。

# 補題 AB, 停止解除みなし申請の定義に必要 (in specialization_a18_AB)
r=Solver()
for k in [年金 A, 年金 B]:
   r.add(ForAll(d,
            Implies(And(k==年金 A, 支給事由発生 (年金 B)(d)),
                And(停止事由発生 (k)(d), 支給 (k)(属月 (d))))))
print(r.check())  # sat   補題 AB は成立
```

```
# f18_specific.py  第十八条  年金の支給期間及び支払期月, 個別版
# k:給付型パラメータ
from essentials import *
from specialization_a18 import * # 通則個別述語対応規則_第十八条

受給権月=(lambda k:期間 (支給事由発生 (k), 受給権利消滅 (k)))
支給=(lambda k:(lambda m:And(受給権月 (k)(m),Not(停止 (k)(m)))))
停止=(lambda k:期間 (停止事由発生 (k), 停止事由消滅 (k)))
年金支払額=(lambda k:(lambda m:
           If(偶数月 (m), 年金支給額 (k)(m-2)+ 年金支給額 (k)(m-1),0)))
年金支給額=(lambda k:(lambda m:If(支給 (k)(m), 年金額 (k)(m),0)))
```

```
# specialization_a18_AB
# 第十八条用通則個別述語結合規則 for 年金 A,B
from honnin_AB import * #年金原簿_年金 A,B
年金 AB,(年金 A, 年金 B)=EnumSort('AB',('A','B'))
# 以下, k:年金 AB, d:日

支給事由発生=(lambda k:(lambda d:
            If(k==年金 A,A 年金受給権利発生 (d),B 年金受給権利発生 (d))))
受給権利消滅=(lambda k:(lambda d:
```

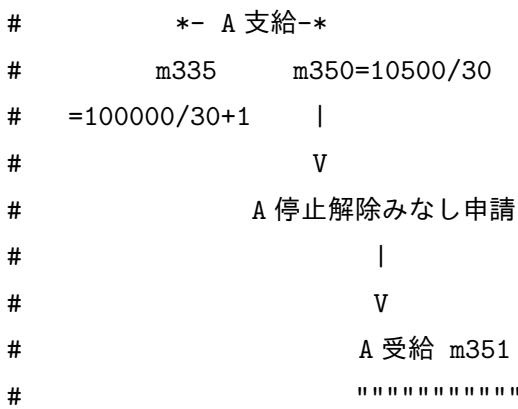
```
            If(k==年金 A,A 年金受給権利消滅 (d),B 年金受給権利消滅 (d))))
停止事由発生=(lambda k:(lambda d:
            If(k==年金 A,B 年金受給権利発生 (d),A 年金受給権利発生 (d))))
停止事由消滅=(lambda k:(lambda d:
            If(k==年金 A,Or(A 年金停止解除申請 (d), 停止解除みなし申請 (年金 A)(d)),
                        Or(B 年金停止解除申請 (d), 停止解除みなし申請 (年金 B)(d)))))
停止解除みなし申請=(lambda k:(lambda d:
            And(k==年金 A, 支給事由発生 (年金 B)(d)))) #補題 AB
```

```
# honnin_AB
# 第二十条併給の調整の検証で使用

#from essentials import *
from z3 import *

A 年金受給権利発生=(lambda d: d==10000)
A 年金受給権利消滅=(lambda d: d==14000)
A 年金停止解除申請=(lambda d: d==12000)
B 年金受給権利発生=(lambda d: d==10500)
B 年金受給権利消滅=(lambda d: d==12000)
B 年金停止解除申請=(lambda d: d==10500)

#         *---------- A -------------------------------------*
#         |           *------ B ----------*                   |
#         |           |                    |                  |
#      A 受給権発生   B 受給権発生          B 受給権利消滅        A 受給権利消滅
#   ------*----------*------------------*------------------*-------
#    d: 10000       10500               12000              14000
#         |          B 支給事由発生
#         |           |    |B 停止事由発生後, 停止解除申請
#         |           |    V
#         |           |   B 受給 m351=10500/30+1
#         |           |  """""""""""""""""""""""
#         V           V
#   A 支給事由発生     A 停止事由発生
#           |         |
```



7.9.4 ノート

- 第二十条では，第一項において，併給不可能な年金給付の組み合わせが述べられており，第二，第三および第四項においては，併給の状況が生じた場合に受給権者が年金を選択するメカニズムが述べられている．ここでは，それに従って選択を行った場合にどのようなことが起こる得るかを検証した．具体的には，第三項が誤りであることが分かる．
- 第一項では併給不可能性が生じるメカニズムは以下のようであると述べられている．既に年金 A の受給権が成立し年金給付の支給を受けている者（A の受給権者）が新たに A と併給不可能な B の受給権も取得すると，A の支給は停止する．この時，第一項における記述が A,B について対称的であることから，A の受給権が既に成立していることにより，(今成立したばかりの）B の受給権による給付も停止され (行われず)，結局，A,B ともに支給されない．
- 受給権者は，第二項で規定されている支給停止解除申請を利用して，停止中の A,B のうちから受給したい年金給付を選択する事ができる．
- ここで問題となるのが第三項の存在である．B の受給権の成立した日の属する月に年金 A が支払われることは，前月に既に決まっているので（第十八条)，第三項により，A に対する解除申請が自動的になされ，従って，次月には A が支給されることになる．よって，もし，当月に B に対する停止解除申請を出すと，次月には B も支給されてしまうことになる．(年金原簿 honnin_AB および付属の図を参照)．
- 第三項がなければ問題は起こらない．なお，現実の運用としては，B の受給権が成立したときに，A か B かの選択をするための選択申出書を提出するようになっており，問題が起こらないようにしている．
- honnin_AB と specialization_a18_AB は，第 I 編で与えたものとは表現は異なるが，内容は同一である．

7.10　第二十六条　支給要件

7.10.1　条文

老齢基礎年金は、保険料納付済期間又は保険料免除期間（第九十条の三第一項の規定により納付することを要しないものとされた保険料に係るものを除く。）を有する者が六十五歳に達したときに、その者に支給する。ただし、その者の保険料納付済期間と保険料免除期間とを合算した期間が十年に満たないときは、この限りでない。

7.10.2　論理式

```
# f26.py　第二十六条　支給要件

from essentials import *
from f5 import 保険料納付済期間,保険料免除期間
m=Int('m')
d1=Int('d1')

老齢基礎年金の支給要件成立=lambda d:\
            And(Exists(m,And(Or(保険料納付済期間(m),保険料免除期間(m)),
                             Not(保険料免除_学生(m)))),
                月数(保険料納付済期間)+月数(保険料免除期間)>=120,
                d==年齢到達日(65,誕生日))
```

7.10.3　検証

```
# v26_ladyA_date.py
# 第二十六条　支給要件
# honnin_ladyA_date の支給要件の確認

import dummy
# 検証で使用する年金原簿モジュール，definitions モジュールを設定
dummy.honnin='honnin_ladyA_date'
dummy.haiguusha='haiguusha_ladyA_date'
dummy.setainushi='setainushi_ladyA_date'
dummy.definitions='definitions_date_3'

import time
```

```
start=time.time()

from essentials import * # z3，年金原簿，用語定義集を import
from f26 import 老齢基礎年金の支給要件成立

start=time.time()
d,d1=Ints('d d1')
s=Solver()
s.add(老齢基礎年金の支給要件成立(d))
print(s.check())
x=s.model()[d].as_long()
print(YMD(x))
print(time.time()-start,"sec")
# sat
# (2020, 5, 9)
# 5.95085883140564 sec  definition_date_3 の時
```

7.10.4 ノート

- 老齢基礎年金の支給要件成立(d)の充足性判定において，変数 d が直接関与するのは年齢が達する日(本人．誕生日,65)(d)のみであるが，他の述語は年金原簿（この場合は，honnin_ladyA）を通して判定結果に関わってくる．すなわち，保険料納付や免除に関する条件は年金原簿に記録されている固定的データから判定され，年齢に関する日付け d についての判定には関係しない．

  これは，保険料納付に関する要件が 60 才前後までに決まってしまい，その後は 65 才になるのを待つという現状とはマッチしている．本来であれば，年金原簿の記録が 60 才までのものであることを確認する必要があるが，現在をそれを行っていない．
- 第二十八条では，年金の受給権者という用語使われているが，これが支給要件とどのような関係であるかは明確には述べられていない．常識的には，支給要件が成立した以後の被保険者をさすもとの考えられる．すなわち，

```
老齢基礎年金の受給権者=(lambda d:
        Exists(d1,And(d1<=d, 老齢基礎年金の支給要件成立(d1))))
```

7.11　第二十七条　年金額

7.11.1　条文

老齢基礎年金の額は、七十八万九百円に改定率（次条第一項の規定により設定し、同条（第一項を除く。）から第二十七条の五までの規定により改定した率をいう。以下同じ。）を乗じて得た額（その額に五十円未満の端数が生じたときは、これを切り捨て、五十円以上百円未満の端数が生じたときは、これを百円に切り上げるものとする。）とする。ただし、保険料納付済期間の月数が四百八十に満たない者に支給する場合は、当該額に、次の各号に掲げる月数を合算した月数（四百八十を限度とする。）を四百八十で除して得た数を乗じて得た額とする。

一　保険料納付済期間の月数

二　保険料四分の一免除期間の月数（四百八十から保険料納付済期間の月数を控除して得た月数を限度とする。）の八分の七に相当する月数

三　保険料四分の一免除期間の月数から前号に規定する保険料四分の一免除期間の月数を控除して得た月数の八分の三に相当する月数

四　保険料半額免除期間の月数（四百八十から保険料納付済期間の月数及び保険料四分の一免除期間の月数を合算した月数を控除して得た月数を限度とする。）の四分の三に相当する月数

五　保険料半額免除期間の月数から前号に規定する保険料半額免除期間の月数を控除して得た月数の四分の一に相当する月数

六　保険料四分の三免除期間の月数（四百八十から保険料納付済期間の月数、保険料四分の一免除期間の月数及び保険料半額免除期間の月数を合算した月数を控除して得た月数を限度とする。）の八分の五に相当する月数

七　保険料四分の三免除期間の月数から前号に規定する保険料四分の三免除期間の月数を控除して得た月数の八分の一に相当する月数

八　保険料全額免除期間（第九十条の三第一項の規定により納付することを要しないものとされた保険料に係るものを除く。）の月数（四百八十から保険料納付済期間の月数、保険料四分の一免除期間の月数、保険料半額免除期間の月数及び保険料四分の三免除期間の月数を合算した月数を控除して得た月数を限度とする。）の二分の一に相当する月数

7.11.2　論理式

```
# f27　第二十七条　年金額
from math import *
from essentials import *
from f5 import 保険料納付済期間, 保険料四分の一免除期間, 保険料半額免除期間,\
                保険料四分の三免除期間, 保険料全額免除_学生を除く期間

保険料納付済月数=月数（保険料納付済期間）
```

保険料四分の一免除月数=月数（保険料四分の一免除期間）
保険料半額免除月数=月数（保険料半額免除期間）
保険料四分の三免除月数=月数（保険料四分の三免除期間）
保険料全額免除_学生を除く月数=月数（保険料全額免除_学生を除く期間）
a1=保険料納付済月数
b2=min(保険料四分の一免除月数,480-a1)
a2=b2*(7/8)
a3=(保険料四分の一免除月数-b2)*(3/8)
b4=min(保険料半額免除月数,480-(保険料納付済月数 + 保険料四分の一免除月数))
a4=b4*(3/4)
a5=(保険料半額免除月数-b4)*(1/4)
b6=min(保険料四分の三免除月数,
        480-(保険料納付済月数 + 保険料四分の一免除月数 + 保険料半額免除月数))
a6=b6*(5/8)
a7=(保険料四分の三免除月数-b6)*(1/8)
a8=min(保険料全額免除_学生を除く月数,
        480-(保険料納付済月数 + 保険料四分の一免除月数 + 保険料半額免除月数
                                                + 保険料四分の三免除月数))*(1/2)
合算月数=a1+a2+a3+a4+a5+a6+a7+a8
減額係数=min(合算月数,480)/480
基準額=780900*改定率
四捨五入前老齢基礎年金額=基準額 if 保険料納付済月数>=480 else 減額係数*基準額
老齢基礎年金額=floor((四捨五入前老齢基礎年金額 +50)/100)*100

7.11.3 検証

```
# v27_date
# 年金原簿 honnin_A_date のもとでの年金額の計算
import dummy
dummy.honnin='honnin_ladyA_date'
dummy.haiguusha='haiguusha_ladyA_date'
dummy.setainushi='setainushi_ladyA_date'
dummy.definitions='definitions_date_3'

import time
start=time.time()
from essentials import *
```

```
from f27 import *

print("保険料納付済月数", 保険料納付済月数)
print("保険料四分の一免除月数", 保険料四分の一免除月数)
print("保険料半額免除月数", 保険料半額免除月数)
print("保険料四分の三免除月数", 保険料四分の三免除月数)
print("保険料全額免除_学生を除く月数", 保険料全額免除_学生を除く月数)
print("合算月数", 合算月数)
print("減額係数", 減額係数)
print("老齢基礎年金額", 老齢基礎年金額)
print(time.time()-start,"sec")

# 保険料納付済月数 432
# 保険料四分の一免除月数 0
# 保険料半額免除月数 12
# 保険料四分の三免除月数 0
# 保険料全額免除月数_学生を除く 0
# 合算月数 441.0
# 減額係数 0.91875
# 老齢基礎年金額 716000
# 3.9343130588531494 sec
```

7.11.4 ノート

- 条文自身は簡単な数式に変換可能であり，論理的な観点からは難解なところはない．f27 の記述内容は、条文で定義されている**老齢基礎年金額**の計算を Python で表現したものであるが，それに使われる基本データ**保険料納付済月数**, **保険料四分の一免除月数**, . . . などは関数「**月数**」を通して充足性判定により得られたものである．**月数**は，**保険料納付済期間**, . . . などの月に関する述語が True になる月を数えるためものである（最大 600 個まで．保険料計算に現れる各述語は，通常は最大 480 月 (40 年間) しか True にならない）．**月数**の定義は，基本用語定義集 defintions_date_3 に与えられている．
- ここで与えた計算式は条文をそのまま定式化したのものであるが，現実問題に適用するには問題がある．条文で定義されている計算式は平成 21 年 4 月以後の保険料免除期間に対するものであり，それ以前の免除期間については附則で別の計算式を使うことが定められている (附則（平成一六年六月一一日法律第一〇四号）第九条，第十条)[2].

  また，細かいことであるが，年金額の 50 円以下の四捨五入に関する記述は現文のままで

は合算月数が 480 以下の場合には適用されないように読める.

- v27_ladyA_date は，ladyA やその配偶者, 世帯主の年金原簿 honnin_ladyA_date, haiguusha_ladyA_date, setainushi_ladyA_date(11) を対象にして，基本用語定義集 definitons_date_3 を用いて ladyA の年金額の計算を行ったものである.

7.12　第二十八条 支給の繰下げ

7.12.1　条文

老齢基礎年金の受給権を有する者であつて六十六歳に達する前に当該老齢基礎年金を請求していなかつたものは、厚生労働大臣に当該老齢基礎年金の支給繰下げの申出をすることができる。ただし、その者が六十五歳に達したときに、他の年金たる給付（他の年金給付（付加年金を除く。）又は厚生年金保険法による年金たる保険給付（老齢を支給事由とするものを除く。）をいう。以下この条において同じ。）の受給権者であつたとき、又は六十五歳に達した日から六十六歳に達した日までの間において他の年金たる給付の受給権者となつたときは、この限りでない。

２　六十六歳に達した日後に次の各号に掲げる者が前項の申出をしたときは、当該各号に定める日において、同項の申出があつたものとみなす。

一　七十歳に達する日前に他の年金たる給付の受給権者となつた者　他の年金たる給付を支給すべき事由が生じた日

二　七十歳に達した日後にある者（前号に該当する者を除く。）　七十歳に達した日

３　第一項の申出をした者に対する老齢基礎年金の支給は、第十八条第一項の規定にかかわらず、当該申出のあつた日の属する月の翌月から始めるものとする。

４　第一項の申出をした者に支給する老齢基礎年金の額は、第二十七条の規定にかかわらず、同条に定める額に政令で定める額を加算した額とする。

7.12.2　論理式

```
# f28.py　第二十八条 支給の繰下げ

# 各種年金に関する受給権が以下の述語として年金原簿 honnin に記録されていると仮定
#　　老齢基礎年金の受給権者, 障害基礎年金の受給権者, 遺族基礎年金の受給権者
#　　老齢を支給事由とするものを除く厚生年金の受給権者, 付加年金の受給権者

from essentials import *

支給繰下げ申出可能=(lambda d:
        And(Not(老齢基礎年金請求日<六十六才に達した日),
            Not(他の年金給付の受給権者 (六十五才に達した日)),
            Not(between(六十五才に達した日 +1, 六十六才に達した日)
                                (他の年金給付の受給権者となった)),
            老齢基礎年金の受給権者 (d)))

支給繰下げ申請みなし日=(lambda d:
```

```
    Implies(And(支給繰下げ申出可能 (支給繰下げ申出日),
                六十六才に達した日<支給繰下げ申出日),
            If(before(七十才に達した日)(他の年金給付の受給権者となった),
               他の年金給付の受給権者となった (d),
               If(七十才に達した日<支給繰下げ申出日,
                  d==七十才に達した日,
                  d==支給繰下げ申出日))))

他の年金給付の受給権者=(lambda d:
      And(Or(障害基礎年金の受給権者 (d),
             遺族基礎年金の受給権者 (d),
             老齢を支給事由とするものを除く厚生年金受給権者 (d)),
          Not(付加年金の受給権者 (d))))

他の年金給付の受給権者となった=成立に至る (他の年金給付の受給権者)
```

7.12.3 検証

```
# v28.py
# honnin_v28, definitons_v28 のもとでの支給の繰下の検証

import dummy
dummy.honnin='honnin_v28'
dummy.definitions='definitions_v28'

from essentials import *
from f28 import * #第二十八条 支給の繰下げ
dd=Int('dd')
s=Solver()
s.add(支給繰下げ申請みなし日 (dd))
print(s.check()) # sat
print(s.model())
# [dd = 6]
```

7.12.4 ノート

- 論理式化にあたっては日付に関するオペレーターを導入することにより，条文の言い回しに近いものにした.

- before(d)(g)：d で指定される日より前に，g が成立する日が存在する．
- between(d1,d2)(h)：d1 以後かつ d2 以前までの間に，h が成立する日が存在する．

- 年金原簿には，受給権者の行う 2 つのアクションが記録されていることを前提とした．それらは，

  （1）老齢基礎年金請求

  （2）支給繰下げ申出

  であり，これらが行われた日付とともに記録されているとする．これらのアクションが受け付けられるための前提条件や順序関係が存在するが，ここでは，アクションが適切に受付けられたことを前提として論理式化を行う．
- 第一項では，繰下げの申出をできる条件が書かれている．「六十六歳に達する前に当該老齢基礎年金を請求していなかつた」が条件であるが，「繰下げ申出と受給請求は排他的である」ことを明記し，「繰下げ申出は六十六歳に達した日以後でしか出来ない」とした方が分かりやすい．
- 第二項では，実際に申出を行った日と，年金額の計算で使われる日（支給繰下げ申請みなし日）との関係が述べられている．
- 第三項，第四項では，繰下げの申出をした場合の年金支給の開始日と年金額が規定されている．年金額は繰下げをした期間の長さに応じて増額されるが，具体的な増加額は政令によるとしている．従って，論理式化は実際に年金計算で使われる繰下げ日 (繰下げ申請見なし日) の算定に関して行った．
- 「支給繰下げ申出可能 ⇒ 支給繰下げ申出」を何処に書くか？　支給繰下げ申出を受付る際には，支給繰下げ申出可能が成立する事が検査されると考えるが，ここでは，繰下げ申請みなし日の決定の条件としてこれを記述した．
- 第二項第一号では，年金給付の「受給権者となったとき」と「支給事由が成立したとき」と区別した表現になっているが，これは同一であり論理式では区別をしていない．
- 条文には、70 歳に達する前に他の年金の受給権者にならずに繰下げ申出をした場合が陽には書かれていない。この場合には、申出をした日が申請見なし日だと考えられる。
- 検証では，年金原簿 honnin_v28 を用いた．年金の支給事由の成立や請求，申出の日にちの順序関係だけが重要なので，1 年=2 日としてモデルを簡単化し，支給繰り下げ申請みなし日が正しく計算されることを確認した．

```
##################
# honnin_v28.py #
##################

# 第二十八条の検証で使用
# 1 年 2 日と単純化し，65 才の誕生日を起点とする通日により日 d を表現
```

```
# 年齢=lambda d:65+d/2

from essentials import *

誕生日=130
六十五才に達した日=年齢到達日(65,誕生日)
六十六才に達した日=年齢到達日(66,誕生日)
七十才に達した日=年齢到達日(70,誕生日)

老齢基礎年金の受給権者=lambda d:d>=0
障害基礎年金の受給権者=lambda d:d>=6
遺族基礎年金の受給権者=lambda d:False
付加年金の受給権者=lambda d:False
老齢を支給事由とするものを除く厚生年金受給権者=lambda d:False
老齢基礎年金請求日=20
支給繰下げ申出日=12

# 上の条件に対する検証結果と条件の順序関係を図示したもの
# sat [dd = 6]
# 年齢  65        66        67        68        69        70          71
# d     0----1----2----3----4----5----6----7----8----9----10----11----12---
#       |                             |                               |
#   老齢支給事由                      障害支給事由                    繰下げ申出
#                                     |
#                                     V
#                                 支給繰下げ申請みなし日

# 以下は，各条件の順序関係などを変化させた場合の結果を染ます.

# 障害基礎年金の受給権者=lambda d:False
# 支給繰下げ申出日=12
# sat [dd = 10]
# 年齢  65        66        67        68        69        70          71
# d     0----1----2----3----4----5----6----7----8----9----10----11----12---
#       |                                                 |           |
#   老齢支給事由                                          |           繰下げ申出
#                                                         V
```

```
#                                                    支給繰下げ申請みなし日

# 障害基礎年金の受給権者=lambda d:False
# 支給繰下げ申出日=7
# sat [dd = 7]
# 年齢  65        66        67        68        69        70          71
# d     0----1----2----3----4----5----6----7----8----9----10----11----12---
#       |                                   |
#   老齢支給事由                            繰下げ申出
#                                           |
#                                           V
#                                         支給繰下げ申請みなし日

# 老齢基礎年金請求=lambda d:d==1
# 支給繰下げ申出=lambda d:d==12
# unsat
# 年齢  65        66        67        68        69        70          71
# d     0----1----2----3----4----5----6----7----8----9----10----11----12---
#       |    |                                                        |
#  老齢支給事由|                                                      繰下げ申出
#           老齢受給請求
#                                       支給繰下げ申請みなし日：なし
```

```
# definitions_v28.py
# v28 において honnin_v28 とともに f28 の検証で使われることを前提

from z3 import *
from essentials import *
d1,d2,d3=Ints('d1 d2 d3')
d=Int('d')

成立に至る=(lambda f:(lambda d: And(Not(f(d-1)),f(d))))
before=lambda d:lambda g:Exists(d1,And(g(d1),d1<d))
between=lambda d1,d2:lambda f:Exists(d,And(f(d),d1<=d,d<=d2))
年齢到達日=lambda age,birth:2*age-birth
```

7.13　第八十九条　保険料免除 (法定免除, 障害, 生活保護, 施設入所)

7.13.1　条文

被保険者（第九十条の二第一項から第三項までの規定の適用を受ける被保険者を除く。）が次の各号のいずれかに該当するに至つたときは、その該当するに至つた日の属する月の前月からこれに該当しなくなる日の属する月までの期間に係る保険料は、既に納付されたものを除き、納付することを要しない。

一　障害基礎年金又は厚生年金保険法に基づく障害を支給事由とする年金たる給付その他の障害を支給事由とする給付であつて政令で定めるものの受給権者（最後に同法第四十七条第二項に規定する障害等級に該当する程度の障害の状態（以下この号において「障害状態」という。）に該当しなくなつた日から起算して障害状態に該当することなく三年を経過した障害基礎年金の受給権者（現に障害状態に該当しない者に限る。）その他の政令で定める者を除く。）であるとき。

二　生活保護法（昭和二十五年法律第百四十四号）による生活扶助その他の援助であつて厚生労働省令で定めるものを受けるとき。

三　前二号に掲げるもののほか、厚生労働省令で定める施設に入所しているとき。

２　前項の規定により納付することを要しないものとされた保険料について、被保険者又は被保険者であつた者（次条から第九十条の三までにおいて「被保険者等」という。）から当該保険料に係る期間の各月につき、保険料を納付する旨の申出があつたときは、当該申出のあつた期間に係る保険料に限り、同項の規定は適用しない。

7.13.2　論理式

```
# f89.py　第八十九条　免除期間　（障害, 生活保護, 施設入所）
from essentials import *

保険料法定免除要件=(lambda m:
            And(期間 3(保険料免除事由)(m),
                Not(Or(保険料四分の三免除 (m),
                       保険料半額免除 (m),
                       保険料四分の一免除 (m),
                       保険料納付済 (m)))))

保険料免除事由=(lambda d:
            Or(障害基礎年金等の受給権者 (d),
               生活保護法の生活扶助を受けている者 (d),
               厚生労働省令施設の入所者 (d)))
```

```
障害基礎年金等の受給権者=(lambda d:
            And(Or(障害基礎年金の受給権者 (d),
                    厚生年金保険法に基づく障害を支給事由とする年金給付の受給者 (d),
                    障害を支給事由とする政令で定める給付の受給権者 (d)),
                        #  国民年金法施行令第六条の五第一項
                Not(Or(And(障害基礎年金の受給権者 (d),
                            最後に障害状態に該当しなくなつた日から三年を経過 (d)),
                        施行令第六条の五第二項で定める者 (d)))))
                        # 国民年金法施行令第六条の五第二項
```

7.13.3　ノート

- 条文第一項第一号中,「現に障害状態に該当しない者に限る」は単なる強調であり，特に意味は無いと解釈した.
- **期間 3(f)**：f が成立する日の属する月の前月から、f が成立しなくなる日の属する月までの月の集合，このような期間が複数ある場合は、それらを合算したものを表す述語．(10)

7.14　第九十条 保険料免除 (申請免除, 全額)

7.14.1　条文

次の各号のいずれかに該当する被保険者等から申請があつたときは、厚生労働大臣は、その指定する期間（次条第一項から第三項までの規定の適用を受ける期間又は学校教育法（昭和二十二年法律第二十六号）第五十条に規定する高等学校の生徒、同法第八十三条に規定する大学の学生その他の生徒若しくは学生であつて政令で定めるもの（以下「学生等」という。）である期間若しくは学生等であつた期間を除く。）に係る保険料につき、既に納付されたものを除き、これを納付することを要しないものとし、申請のあつた日以後、当該保険料に係る期間を第五条第三項に規定する保険料全額免除期間（第九十四条第一項の規定により追納が行われた場合にあつては、当該追納に係る期間を除く。）に算入することができる。ただし、世帯主又は配偶者のいずれかが次の各号のいずれにも該当しないときは、この限りでない。

一　当該保険料を納付することを要しないものとすべき月の属する年の前年の所得（一月から厚生労働省令で定める月までの月分の保険料については、前々年の所得とする。以下この章において同じ。）が、その者の扶養親族等の有無及び数に応じて、政令で定める額以下であるとき。

二　被保険者又は被保険者の属する世帯の他の世帯員が生活保護法による生活扶助以外の扶助その他の援助であつて厚生労働省令で定めるものを受けるとき。

三　地方税法（昭和二十五年法律第二百二十六号）に定める障害者であつて、当該保険料を納付することを要しないものとすべき月の属する年の前年の所得が政令で定める額以下であるとき。

四　地方税法に定める寡婦であつて、当該保険料を納付することを要しないものとすべき月の属する年の前年の所得が前号に規定する政令で定める額以下であるとき。

五　保険料を納付することが著しく困難である場合として天災その他の厚生労働省令で定める事由があるとき。

２　前項の規定による処分があつたときは、年金給付の支給要件及び額に関する規定の適用については、その処分は、当該申請のあつた日にされたものとみなす。

３　第一項の規定による処分を受けた被保険者から当該処分の取消しの申請があつたときは、厚生労働大臣は、当該申請があつた日の属する月の前月以後の各月の保険料について、当該処分を取り消すことができる。

４　第一項第一号、第三号及び第四号に規定する所得の範囲及びその額の計算方法は、政令で定める。

7.14.2　論理式

```
# f90.py　第九十条 保険料免除（全額, 申請免除）
from essentials import *

保険料全額免除要件=(lambda m:
```

```
            And(Not(Or(保険料四分の三免除(m),
                       保険料半額免除(m),
                       保険料四分の一免除(m),
                       保険料納付済(m),
                       学生等(m))),
                本人世帯主配偶者が経済的困窮(m)))

経済的困窮=(lambda p:lambda m:
            Or(p.前年の所得が政令第六条の七で定める額以下(m),
               p.生活保護以外の厚生労働省令で定める援助を受給(m),
               p.障害者であり前年の所得が政令で定める額以下(m),
               p.寡婦であり前年の所得が政令で定める額以下(m),
               p.天災など省令により保険料納付が著しく困難(m)))

# 以下に於いて，本人，世帯主，配偶者は，それぞれの年金原簿を表し，essentials において
# import される.

本人世帯主配偶者が経済的困窮=(lambda m:
            And(経済的困窮(本人)(m),
                Implies(世帯主が本人以外(m),経済的困窮(世帯主)(m)),
                Implies(配偶者がいる(m),経済的困窮(配偶者)(m))))

########
# 以下を f90 に加えることも可能である．しかし，内容が細部に亘るので，事前にその評価を行い，
# その結果によって honnin を変更してから f90 に入る方が良いと思われる.

# import seirei6_7 as 政令第六条の七
#
# 前年の所得が政令第六条の七で定める額以下=lambda m:\
#              前年度の所得(m) <= 政令第六条の七.定める額
# 前年度の所得=lambda m: If(And(1<=m,m<=省令で定める月),
#                           所得(属年(m)-2),所得(属年(n)-1))

######
# seirei6_7.py
# 定める額=35*(本人.扶養親族の数+1)+22
```

7.14.3　ノート

- 被保険者が，月 m において「保険料全額免除」となるための要件が定義されている．これは，被保険者の行う保険料全額免除の申請が認められ，月 m が保険料免除期間 (第五条) に算入されるための要件を定めている．算入される月は，具体的には，年金原簿に免除として記録される．

  免除申請 ⇒ 免除要件成立を確認 ⇒ 免除期間に算入 ⇒ 年金原簿に免除として記録

  条文の論理式化では，f89,90,90_2,f90_3 などで免除要件が定義され，f5 において免除期間が定義されている．免除申請と年金原簿への記録は適切に行われるとして形式化は行っていない．また，条文では，複数の月からなる期間がまとめて免除される場合を述べているが，論理式では簡単のために単一の月を対象とした．
- ここでの全額免除は被保険者の申請によるものであり，第八十九条で定義される法定の全額免除とは別のものである．申請による保険料免除はこれ以外に，四分の一，半額，四分の三，学生特例がある．附則には若年者納付猶予が規定されている．
- 申請が承認されるための要件は，基本的には，（１）他の一部免除などと重複がないこと，（２）本人・世帯主・配偶者全てが経済的に困窮していることである．
- これらの条件の詳細は，別途国民年金法施行令と施行規則に述べられている．
  - 施行令第六条の六：学生等の詳細
  - 施行令第六条の七，八：経済的困窮を定める基準金額
  - 施行規則七十七条，七十七条の二：申請書の内容，期間

  これらの施行令や施行規則に記述されている内容は，必要であれば論理式化可能であると考える．
- 経済的困窮は，本人のみならず，世帯主や配偶者についても述べる必要がある．本稿での論理式化の方法は，「特定の個人によらない記述」＋「個人の年金原簿」の形で行われている．殆どの条文は関連する個人は一人であり，条文の検証ではその人（「本人」と呼んでいる）の年金原簿について行われる．
- これに対し，本条文では，複数の人の年金原簿を参照する必要がある．論理式記述では，本人，配偶者，世帯主の３つの年金原簿に対して，それぞれの経済的困窮を判定し，その結果から**本人世帯主配偶者が経済的困窮**を決定している．述語**経済的困窮**は，個人を表す変数 p を持ち，それには本人，配偶者，世帯主の年金原簿モジュール**本人，配偶者，世帯主**が渡される．個人をパラメータ化するこのメカニズムにより論理式記述はコンパクトになった．

  ```
  本人世帯主配偶者が経済的困窮=(lambda m:
              And(経済的困窮（本人)(m),
                  Implies(世帯主が本人以外（m),経済的困窮（世帯主)(m)),
                  Implies(配偶者がいる（m),経済的困窮（配偶者)(m))))
  ```
- 第一項では追納が行われた期間が免除の対象にならないことが特に述べられているが，追納

は正常な納付と区別されずに（第九十四条）年金原簿に述語**保険料納付済**として記録されるので，論理式化では考慮する必要は無い．

- 第三項では，免除申請が認められた後で取り消しの請求を出せることが述べられている．これと追納との関係が明確でないが，単なる取り消しであれば，年金原簿を前の状態に復旧することにより実現できる．（十分には理解できていない．TAC にも解説されていない．多分，免除期間が終わらないうちなら免除の取り消しが可能で，取り消しの申請の前月以後から正常な納付に戻ることが可能であるのかも知れない．）

7.15　第九十条の二　保険料免除 (申請免除, 一部)

7.15.1　条文

次の各号のいずれかに該当する被保険者等から申請があつたときは、厚生労働大臣は、その指定する期間（前条第一項若しくは次項若しくは第三項の規定の適用を受ける期間又は学生等である期間若しくは学生等であつた期間を除く。）に係る保険料につき、既に納付されたものを除き、その四分の三を納付することを要しないものとし、申請のあつた日以後、当該保険料に係る期間を第五条第四項に規定する保険料四分の三免除期間（第九十四条第一項の規定により追納が行われた場合にあつては、当該追納に係る期間を除く。）に算入することができる。ただし、世帯主又は配偶者のいずれかが次の各号のいずれにも該当しないときは、この限りでない。

一　当該保険料を納付することを要しないものとすべき月の属する年の前年の所得が、その者の扶養親族等の有無及び数に応じて、政令で定める額以下であるとき。

二　前条第一項第二号から第四号までに該当するとき。

三　保険料を納付することが著しく困難である場合として天災その他の厚生労働省令で定める事由があるとき。

２　次の各号のいずれかに該当する被保険者等から申請があつたときは、厚生労働大臣は、その指定する期間（前条第一項若しくは前項若しくは次項の規定の適用を受ける期間又は学生等である期間若しくは学生等であつた期間を除く。）に係る保険料につき、既に納付されたものを除き、その半額を納付することを要しないものとし、申請のあつた日以後、当該保険料に係る期間を第五条第五項に規定する保険料半額免除期間（第九十四条第一項の規定により追納が行われた場合にあつては、当該追納に係る期間を除く。）に算入することができる。ただし、世帯主又は配偶者のいずれかが次の各号のいずれにも該当しないときは、この限りでない。

一　当該保険料を納付することを要しないものとすべき月の属する年の前年の所得が、その者の扶養親族等の有無及び数に応じて、政令で定める額以下であるとき。

二　前条第一項第二号から第四号までに該当するとき。

三　保険料を納付することが著しく困難である場合として天災その他の厚生労働省令で定める事由があるとき。

３　次の各号のいずれかに該当する被保険者等から申請があつたときは、厚生労働大臣は、その指定する期間（前条第一項若しくは前二項の規定の適用を受ける期間又は学生等である期間若しくは学生等であつた期間を除く。）に係る保険料につき、既に納付されたものを除き、その四分の一を納付することを要しないものとし、申請のあつた日以後、当該保険料に係る期間を第五条第六項に規定する保険料四分の一免除期間（第九十四条第一項の規定により追納が行われた場合にあつては、当該追納に係る期間を除く。）に算入することができる。ただし、世帯主又は配偶者のいずれかが次の各号のいずれにも該当しないときは、この限りでない。

一　当該保険料を納付することを要しないものとすべき月の属する年の前年の所得が、その者の扶養親族等の有無及び数に応じて、政令で定める額以下であるとき。

二　前条第一項第二号から第四号までに該当するとき。
三　保険料を納付することが著しく困難である場合として天災その他の厚生労働省令で定める事由があるとき。
４　前条第三項の規定は、前三項の規定による処分を受けた被保険者から当該処分の取消しの申請があつたときに準用する。
５　第一項第一号、第二項第一号及び第三項第一号に規定する所得の範囲及びその額の計算方法は、政令で定める。
６　第一項から第三項までの規定により納付することを要しないものとされたその一部の額以外の残余の額に五円未満の端数が生じたときは、これを切り捨て、五円以上十円未満の端数が生じたときは、これを十円に切り上げるものとする。

7.15.2　論理式

```
# f90_2　第九十条の二　保険料免除（一部，申請免除）
from essentials import *

保険料四分の三免除要件=(lambda m:
            And(Not(Or(保険料全額免除（m），
                       保険料半額免除（m），
                       保険料四分の一免除（m），
                       保険料納付済（m），
                       学生等（m））），
                本人世帯主配偶者が経済的困窮 1(m)))

保険料半額免除要件=(lambda m:
            And(Not(Or(保険料全額免除（m），
                       保険料四分の三免除（m），
                       保険料四分の一免除（m），
                       保険料納付済（m），
                       学生等（m））），
                本人世帯主配偶者が経済的困窮 1(m)))

保険料四分の一免除要件=(lambda m:
            And(Not(Or(保険料全額免除（m），
                       保険料四分の三免除（m），
                       保険料半額免除（m），
                       保険料納付済（m），
```

```
                学生等(m))),
            本人世帯主配偶者が経済的困窮 2(m)))

経済的困窮 1=(lambda p:lambda m:
        Or(p.前年の所得が政令第六条の八で定める額以下(m),
           p.生活保護以外の厚生労働省令で定める援助を受給(m),
           p.障害者であり前年の所得が政令で定める額以下(m),
           p.寡婦であり前年の所得が政令で定める額以下(m),
           p.天災など省令により保険料納付が著しく困難(m)))

本人世帯主配偶者が経済的困窮 1=(lambda m:
        And(経済的困窮 1(本人)(m),
            Implies(世帯主が本人以外(m),経済的困窮 1(世帯主)(m)),
            Implies(配偶者がいる(m),経済的困窮 1(配偶者)(m))))

経済的困窮 2=(lambda p:lambda m:
        Or(p.前年の所得が政令第六条の九で定める額以下(m),
           p.生活保護以外の厚生労働省令で定める援助を受給(m),
           p.障害者であり前年の所得が政令で定める額以下(m),
           p.寡婦であり前年の所得が政令で定める額以下(m),
           p.天災など省令により保険料納付が著しく困難(m)))

本人世帯主配偶者が経済的困窮 2=(lambda m:
        And(経済的困窮 2(本人)(m),
            Implies(世帯主が本人以外(m),経済的困窮 2(世帯主)(m)),
            Implies(配偶者がいる(m),経済的困窮 2(配偶者)(m))))
```

7.15.3 検証

```
# v90_2
# 異なる種別の免除期間が同じ月に関しては両立しないことを
# 年金原簿 honnin_ladyA_date について確認したもの

import dummy
dummy.honnin='honnin_ladyA_date' # 本人
dummy.setainushi='setainushi_ladyA_date' # 世帯主
dummy.haiguusha='haiguusha_ladyA_date' # 配偶者
```

```
dummy.definitions='definitions_date_3'

from essentials import *
from f5 import *

m=Int('m')
s=Solver()
p=And(保険料四分の一免除期間(m),保険料半額免除期間(m))
s.add(p)
print(s.check())
# unsat
```

7.15.4 ノート

- ここでは，被保険者の申請により保険料の一部（四分の一，半分，四分の三）の納付が免除される要件が定義されている．他の納付免除と重複せず，免除された以外の残りの保険料を納付し，かつ，本人，配偶者，世帯主がいずれも経済的に困窮していることが納付免除の要件である．申請が認められると，年金原簿には，その月が該当免除となることと，残りの納付が行われたことが記録され，その月は第五条のおいて該当免除の期間に算定される．
- 納付免除のカテゴリーが重ならないことを，年金原簿 honnin_ladyA にもとづいて確認すると同時に，より一般的な条件のもとで検証した．

7.16　第九十条の三　保険料免除 (申請免除, 学生)

7.16.1　条文

次の各号のいずれかに該当する学生等である被保険者又は学生等であつた被保険者等から申請があつたときは、厚生労働大臣は、その指定する期間（学生等である期間又は学生等であつた期間に限る。）に係る保険料につき、既に納付されたものを除き、これを納付することを要しないものとし、申請のあつた日以後、当該保険料に係る期間を第五条第三項に規定する保険料全額免除期間（第九十四条第一項の規定により追納が行われた場合にあつては、当該追納に係る期間を除く。）に算入することができる。

一　当該保険料を納付することを要しないものとすべき月の属する年の前年の所得が、その者の扶養親族等の有無及び数に応じて、政令で定める額以下であるとき。

二　第九十条第一項第二号から第四号までに該当するとき。

三　保険料を納付することが著しく困難である場合として天災その他の厚生労働省令で定める事由があるとき。

２　第九十条第二項の規定は、前項の場合に準用する。

３　第一項第一号に規定する所得の範囲及びその額の計算方法は、政令で定める。

7.16.2　論理式

```
# f90_3_1　第九十条の三　免除（学生）
from essentials import *

保険料免除_学生要件=(lambda m:
            And(学生等 (m),
                Not(保険料納付済 (m)),
                本人が経済的困窮 (m)))

本人が経済的困窮=(lambda m:
            Or(前年の所得が政令第六条の八で定める額以下 (m),
               生活保護以外の厚生労働省令で定める援助を受給 (m),
               障害者であり前年の所得が政令で定める額以下 (m),
               寡婦であり前年の所得が政令で定める額以下 (m),
               天災など省令により保険料納付が著しく困難 (m)))
```

7.17　附則（平成一六年六月一一日法律第一〇四号）第十九条

7.17.1　条文

第十九条　平成十七年四月から平成十八年六月までの期間において、三十歳に達する日の属する月の前月までの被保険者期間がある第一号被保険者等（国民年金法第七条第一項第一号に規定する第一号被保険者又は第一号被保険者であった者をいう。以下この条において同じ。）であって次の各号のいずれかに該当するものから申請があったときは、厚生労働大臣は、当該被保険者期間のうちその指定する期間（第二条の規定による改正後の国民年金法第九十条第一項若しくは第九十条の二第一項の規定の適用を受ける期間又は同法第九十条第一項に規定する学生等（以下「学生等」という。）である期間若しくは学生等であった期間を除く。）に係る国民年金の保険料については、国民年金法第八十八条第一項の規定にかかわらず、既に納付されたもの及び同法第九十三条第一項の規定により前納されたものを除き、これを納付することを要しないものとし、申請のあった日以後、当該保険料に係る期間を同法第五条第四項に規定する保険料全額免除期間（同法第九十四条第一項の規定により追納が行われた場合にあっては、当該追納に係る期間を除く。）に算入することができる。ただし、配偶者が次の各号のいずれにも該当しないときは、この限りでない。

一　当該保険料を納付することを要しないものとすべき月の属する年の前年の所得（一月から厚生労働省令で定める月までの月分の保険料については、前々年の所得とする。）が、その者の所得税法（昭和四十年法律第三十三号）に規定する控除対象配偶者及び扶養親族の有無及び数に応じて、政令で定める額以下であるとき。

二　第二条の規定による改正後の国民年金法第九十条第一項第二号から第四号までに該当するとき。

三　国民年金の保険料を納付することが著しく困難である場合として天災その他の厚生労働省令で定める事由があるとき。

２　平成十八年七月から平成三十七年六月までの期間において、三十歳に達する日の属する月の前月までの被保険者期間がある第一号被保険者等であって次の各号のいずれかに該当するものから申請があったときは、厚生労働大臣は、当該被保険者期間のうちその指定する期間（第四条の規定による改正後の国民年金法第九十条第一項若しくは第九十条の二第一項から第三項までの規定の適用を受ける期間又は学生等である期間若しくは学生等であった期間を除く。）に係る国民年金の保険料については、国民年金法第八十八条第一項の規定にかかわらず、既に納付されたものを除き、これを納付することを要しないものとし、申請のあった日以後、当該保険料に係る期間を同法第五条第四項に規定する保険料全額免除期間（第四条の規定による改正後の国民年金法第九十四条第一項の規定により追納が行われた場合にあっては、当該追納に係る期間を除く。）に算入することができる。ただし、配偶者が次の各号のいずれにも該当しないときは、この限りでない。

一　当該保険料を納付することを要しないものとすべき月の属する年の前年の所得（一月から厚生労働省令で定める月までの月分の保険料については、前々年の所得とする。）が、その者の所得税法に規定する同一生計配偶者及び扶養親族の有無及び数に応じて、政令で定める額以下であるとき。

二　第四条の規定による改正後の国民年金法第九十条第一項第二号から第四号までに該当するとき。
三　国民年金の保険料を納付することが著しく困難である場合として天災その他の厚生労働省令で定める事由があるとき。
３　国民年金法第九十条第二項及び第三項の規定は、前二項の場合に準用する。
４　第一項又は第二項の規定により保険料を納付することを要しないものとされた者及びこれらの規定により納付することを要しないものとされた保険料については、国民年金法その他の法令の規定を適用する場合においては、同法第九十条の三第一項の規定により保険料を納付することを要しないものとされた者及び同項の規定により納付することを要しないものとされた保険料とみなすほか、これらの規定の適用に関し必要な事項は、政令で定める。
５　国民年金法附則第五条第一項の規定による被保険者については、第一項及び第二項の規定を適用しない。
６　第一項第一号及び第二項第一号に規定する所得の範囲及びその額の計算方法は、政令で定める。

7.17.2　論理式

```
# fH16-104-19　附則平成一六年六月一一日法律第一〇四号第十九条（若年納付猶予）
# 現在の保険料免除に関する f89,f90,f90_2,f90_3 には考慮されてはいない.

from f11 import 第一号被保険者期間
from essentials import *
m=Int('m')

三十才に達する日の前月=前月（年齢到達（30，誕生日））
平成 17 年 4 月から平成 18 年 6 月まで=MM((平成（17),4),(平成（18),6))
平成 18 年 7 月から平成 37 年 6 月まで=MM((平成（18),7),(平成（37),6))

保険料納付猶予_若年=lambda m:Or(保険料納付猶予_若年 1(m),
                                 保険料納付猶予_若年 2(m))
保険料納付猶予_若年 1=(lambda m: # m:納付猶予対象月
          And(若年 1,
              Not(Or(本人．保険料全額免除（m),
                     本人．保険料半額免除（m),
                     本人．学生等（m),
                     本人．保険料納付済（m),
                     本人．保険料前納（m))),
              本人配偶者が経済的困窮（m)))
```

```
保険料納付猶予_若年 2=(lambda m:
            And(若年 2,
                Not(Or(本人.保険料全額免除 (m),
                       本人.保険料四分の三免除 (m),
                       本人.保険料半額免除 (m),
                       本人.保険料四分の一免除 (m),
                       本人.学生等 (m),
                       本人.保険料納付済 (m))),
                本人配偶者が経済的困窮 (m)))

若年 1=Exists(m, # m:納付猶予申請月
            And(平成 17 年 4 月から平成 18 年 6 月まで (m),
                三十才に達する日の前月 (m),
                第一号被保険者期間 (m)))
若年 2=Exists(m,
            And(平成 18 年 7 月から平成 37 年 6 月まで (m),
                三十才に達する日の前月 (m),
                第一号被保険者期間 (m)))

本人配偶者が経済的困窮=(lambda m:
            And(経済的困窮 (本人)(m),
                Implies(本人.配偶者がいる (m), 経済的困窮 (配偶者)(m))))
経済的困窮=(lambda p:(lambda m:
            Or(p.前年の所得が政令_施行令第六条の七で定める額以下 (m),
               p.生活保護以外の厚生労働省令で定める援助を受給 (m),
               p.障害者であり前年の所得が政令で定める額以下 (m),
               p.寡婦であり前年の所得が政令で定める額以下 (m),
               p.天災などにより保険料納付が著しく困難 (m))))
```

7.17.3 検証

```
# fH16_104_19.py　附則（平成一六年六月一一日法律第一〇四号）第十九条　若年納付猶予の検証
# 年金原簿 honnin_ladyB_H16_104_19 のもとで，保険料納付猶予_若年 1 を満たす月を確認
# 配偶者の年金原簿は，本人のもので代用した.

import dummy
```

```
dummy.honnin='honnin_ladyB_H16_104_19' # 本人
dummy.haiguusha='honnin_ladyB_H16_104_19' # 配偶者：本人で代用
dummy.definitions='definitions_date_3'

import time
start=time.time()
from essentials import *
from fH16_104_19 import *
y,m,m1=Ints('y m m1')

s=Solver()
s.add(保険料納付猶予_若年1(m))
print(s.check())
print(s.model()) # [m = 1757]
solve(s.model()[m]==通月(平成(y),m1),1<=m1,m1<=12) # y=17,m1=4
# 平成17年4月
print(充足リスト(保険料納付猶予_若年1,50))
# [1757, 1758, 1759, 1760, 1761, 1762, 1763, 1764, 1765, 1766, 1767, 1768]
# 平成17年4月～平成18年3月
```

7.17.4 ノート

- この附則は若年者に対して学生と同様な保険料全額免除 (納付猶予) 規則を定めるものである．この条文は改正法に付随する附則のなかにあり，その効果が「平成十七年四月から平成十八年六月までの期間」に限られる経過措置である．３０歳に達する日の前月がこの期間にある第一号被保険者のみを対象とする保険料免除である．これは論理式化に於いては，**若年1**と**若年2**に分けて表現されている．**若年1**は，平成17年4月から平成18年6月までの間において，30才未満の第一号被保険者を対象としており，**若年2**は平成18年7月から平成37年6月までを対象にしている．
- 納付猶予要件が２つに分かれている理由は，適用される国民年金法が平成18年7月から変わったためである．

  すなわち，平成18年6月までのもの「第二条の規定による改正後の国民年金法」では，保険料納付免除（申請）には全額，半額と学生特例しかなかったのに反し，平成18年7月からのもの「第四条の規定による改正後の国民年金法」は，新たに四分の三と四分の一免除が追加されている．若年者猶予は当然他の免除と両立はできないので，これらの２つの国民年金法の適用を受ける被保険者を区別し記述を明確化するために第一項と二項に分けて条文が記述されている．これに合わせ，論理式化では納付猶予要件を，**保険料納付猶予_若年要**

件 1 と保険料納付猶予_若年要件 2 に分けている．

なお，本条の追加により本則で規定されている他の免除要件においても，保険料納付猶予_若年と両立しないことを追加する必要があるが，これは現在の論理式化では行っていない．

- 保険料前納の場合の扱いが要件 1 と要件 2 で若干異なっている．要件 1 では，過去に月々支払った「納付済み」保険料と，纏めて一括して「前納」された保険料ともに免除の対象にならないとしているのに反し，要件 2 では通常の納付済みのみが対象にならないとしている．既に支払った保険料の扱いが「第二条の規定による改正後の国民年金法」「第四条の規定による改正後の国民年金法」において同じ（納付済み，前納ともに免除対象でない）であることを考えると，この附則における扱いの違いがどこから来るのかが分からない．なお，関連する 3 つの法令の施行時期は以下のようである．
  - 第二条の規定による改正後の国民年金法：平成１７年４月１日
  - 第四条の規定による改正後の国民年金法：平成１８年６月１日
  - 附則（平成一六年六月一一日法律第一〇四号）：平成１７年７月１日

  どこかの時点で方針の変更がなされたと思われるが，残念ながら特定することは出来ていない．
- 本人や配偶者が経済的に困窮していることが免除条件であるが，その第一号は第一項と第二項では少し異なっているが，これについては精査していない．両方で同じものを用いている．
- なお，国民年金法は平成 16 年に大きな改正を行い，平成 16 年法律第 104 号 (国民年金法の一部を改正する法律) として立法化された．この法律自身は，衆議院の HP
  http://www.shugiin.go.jp/internet/itdb_housei.nsf/html/housei/15920040611104.htm
  から見ることができる．また，改正された法律の新旧対象条文は厚労省 HP
  https://www.mhlw.go.jp/topics/bukyoku/nenkin/nenkin/kaisei-taishou.html
  にあり，そこから「第二条の規定による改正後の国民年金法」「第四条の規定による改正後の国民年金法」を知ることができる．
- 他の納付免除と同様に，納付猶予要件は単一の月についてと簡単化してある．
- 平成 26 年の改正により，猶予は 50 才未満にまで拡大されたが，申請書類などこれに関する具体的な情報は，以下に年金機構の情報がある．
  https://www.nenkin.go.jp/shinsei/kokunen/kokunen.files/20.pdf

7.18　第九十四条　保険料の追納

　第九十四条　被保険者又は被保険者であつた者（老齢基礎年金の受給権者を除く。）は、厚生労働大臣の承認を受け、第八十九条第一項、第九十条第一項又は第九十条の三第一項の規定により納付することを要しないものとされた保険料及び第九十条の二第一項から第三項までの規定によりその一部の額につき納付することを要しないものとされた保険料（承認の日の属する月前十年以内の期間に係るものに限る。）の全部又は一部につき追納をすることができる。ただし、同条第一項から第三項までの規定によりその一部の額につき納付することを要しないものとされた保険料については、その残余の額につき納付されたときに限る。

２　前項の場合において、その一部につき追納をするときは、追納は、第九十条の三第一項の規定により納付することを要しないものとされた保険料につき行い、次いで第八十九条第一項若しくは第九十条第一項の規定により納付することを要しないものとされた保険料又は第九十条の二第一項から第三項までの規定によりその一部の額につき納付することを要しないものとされた保険料につき行うものとし、これらの保険料のうちにあつては、先に経過した月の分から順次に行うものとする。ただし、第九十条の三第一項の規定により納付することを要しないものとされた保険料より前に納付義務が生じ、第八十九条第一項若しくは第九十条第一項の規定により納付することを要しないものとされた保険料又は第九十条の二第一項から第三項までの規定によりその一部の額につき納付することを要しないものとされた保険料があるときは、当該保険料について、先に経過した月の分の保険料から追納をすることができるものとする。

３　第一項の場合において追納すべき額は、当該追納に係る期間の各月の保険料の額に政令で定める額を加算した額とする。

４　第一項の規定により追納が行われたときは、追納が行われた日に、追納に係る月の保険料が納付されたものとみなす。

５　前各項に定めるもののほか、保険料の追納手続その他保険料の追納について必要な事項は、政令で定める。

7.18.1　論理式

```
# f94.py　第九十四条　保険料の追納

from essentials import *
m,m1=Ints('m m1')
保険料免除種別,(法定, 全額, 四分の三, 半額, 四分の一, 学生)= \
        EnumSort('保険料免除種別',
                 ('法定','全額','四分の三','半額','四分の一','学生'))
e=Const('e', 保険料免除種別)
```

```
# 追納可能性判定
# m0:追納請求をした月,m1:追納対象の月,e:保険料免除の種別

追納可能=(lambda m0,m1,e:
            And(m0-m1<=120,
                Or(And(e!=学生, 追納対象保険料免除 (本人)(e)(m1),
                       ForAll(m,Implies(本人. 保険料免除 (m),m1<=m))),
                   And(e==学生, 本人. 保険料免除_学生 (m1),
                       ForAll(m,Implies(本人. 保険料免除_学生 (m),m1<=m))))))
# p; 年金原簿
追納対象保険料免除=(lambda p:lambda e:lambda m:
            If(e==法定,p. 保険料法定免除 (m),
            If(e==全額,p. 保険料全額免除 (m),
            If(e==四分の三,p. 保険料四分の三免除 (m),
            If(e==半額,p. 保険料半額免除 (m),
            If(e==四分の一,p. 保険料四分の一免除 (m),
            If(e==学生,p. 保険料免除_学生 (m),False)))))))

追納額=(lambda e,m0,m1:
             保険料 (e,m1)+ 政令により定まる加算額 (e,m0,m1))

# 追納可能 (m0,m1,e) が True となり追納が行われた場合には, 追納前の年金原簿「本人」と
# 追納後の年金原簿「本人_post」の間には次の P,Q,R が成立しなければならない.

P = 本人_post. 保険料納付済 (m)== \
           If(m==m1,True, 本人. 保険料納付済 (m))
Q = 本人_post. 追納により納付とみなされる期間 (m)== \
           If(m==m1,True, 本人. 追納により納付とみなされる期間 (m))
R = 追納対象保険料免除 (本人_post)(e)(m)== \
           If(m==m1,False, 追納対象保険料免除 (本人)(e)(m))
```

7.18.2 検証

```
# v94.py
# 第九十四条保険料の追納の年金原簿 honnin_ladyA_date のもとでのテスト

import dummy
```

```
dummy.honnin='honnin_ladyA_date'
dummy.honnin_post='honnin_ladyA_post'
dummy.definitions='definitions_date_3'

import time
start=time.time()
from essentials import * # honnin_ladyA
from f94 import *
m,m1=Ints('m m1')
e=Const('e', 保険料免除種別)

s=Solver()
s.add(追納可能 (通月 (1980,10),m, 学生))
print(s.check()) # sat
print(YM(s.model()[m].as_long())) # (1975.0, 4)
# (1975,4) は honnin_ladyA_date において，保険料免除_学生が True となる最初の月

u=Solver()
u.add(ForAll([m,m1,e],Implies(追納可能 (m,m1,e),And(P,Q,R))))
print(u.check())
# sat  本人, 本人_post は P,Q,R を満たしている.

# 追納申請日を変えると結果は異なる.
r=Solver()
r.add(追納可能 (通月 (2010,1),m, 学生))
print(r.check())
# unsat  西暦 2010 年 1 月以降では，保険料免除_学生はない.

t=Solver()
t.add(追納可能 (通月 (2010,1),m, 半額))
print(t.check()) # sat
print(YM(t.model()[m].as_long())) # (2007.0, 2)
# (2007,2) は半額免除となる最初の月

print(time.time()-start,'sec')
# 0.26294398307800293 sec
```

7.18.3 ノート

- 追納の記述には，（１）追納申請の記述と（２）追納による年金原簿の変更が含まれる．ある月 m の追納申請が承認されるには，追納の請求を行った月から１０年以内の月で，免除された中で一番古いものか，あるいは，学生特例による免除の中で一番古いものである（第一項)．なお，第一項中，「ただし」以前では学生特例を優先する記述になっているが，「ただし」以降では学生特例優先が除かれている．また，年金機構の HP では，最も古いものからの追納のみが書かれている．一方，追納による年金原簿の変更は，追納された月 m が免除から納付に変更されることである（第四項)．
- f94 は，追納申請の承認可能性および追納による年金原簿の変更に関する記述である．**追納可能** (m0,m1,e) は，種別 e，月 m1 の免除の追納が月 m0 に申請できる条件を記述したものである．また，追納申請が受理されたときには，その結果が年金原簿に反映されなければならないが，それは追納後の年金原簿**本人**_post と追納前の年金原簿**本人**の関係を表す命題 P,Q,R によって形式化される．
- v94 では，年金原簿 honnin_ladyA のもとでのいくつかの事例について，**追納可能**の定義の正しさについて確認を行った．
- これまで年金原簿には保険料納付や保険料免除に関する情報が月単位で記録されてきたが，厳密に第四項に対応するには納付の日を記録することが必要になる．しかしながら，最終的には，年金額や年金支給要件の決定には月単位の記録のみで十分であるので，問題が生じるまでは現状のままにしておく．

8　日付に関する形式化と抽象日付けデータを用いた検証

8.1　日付における西暦の使用

- 国民年金法の条文の論理式記述に現れる殆どの述語は日付に関するものである．また，今回記述を行った殆どの条文に関しては，具体の日付は現れず，日，月，年と言った日付の基本的構造のみが意味があったので，論理式化の検証に関しては，実際の暦に基づく日付を用いず，条文毎にその検証に必要な範囲内で簡略化した人工的な日付けによる年金原簿を用いることが多かった．年金原簿 `honnin_v7` では，1 月 30 日，1 年 360 日とし，また，第二十八条の検証で用いた `honnin_v28` では，65 才に達した日を基点として 1 年 2 日という極端に人工的な日付けを用いている．
- しかしながら，例えば，若年納付猶予に関する条文，附則平成一六年六月一一日法律第一〇四号第十九条では，若年要件の記述に具体の日付けが現れ，その記述には実際の暦による日付けの形式化が必要になる．また，条文の論理式化をもとに定理証明器を利用した年金システムシミュレーションを現実のデータに対して行う場合には，年金原簿の記述に具体の日付けによるものを用いる必要がある．
- 西暦 1858 年 11 月 17 日を原点としてそこからの日数を修正ユリウス通日 (Modified Jurian Date) というが，ウィキペディアによると，西暦 (y, m, d) とその修正ユリウス通日 mjd との間には次の相互変換式が成立する [3]．ただし 1 月、2 月は前年の 13 月、14 月として計算する．

$$mjd = \lfloor 365.25y \rfloor + \lfloor y/400 \rfloor - \lfloor y/100 \rfloor + \lfloor 30.59(m-2) \rfloor + d - 67891233$$

$$\begin{aligned} n &= mjd + 678881 \\ a &= 4n + 3 + 4 \left\lfloor \frac{3}{4} \left\lfloor \frac{4(n+1)}{146\ 097} + 1 \right\rfloor \right\rfloor \\ b &= 5 \left\lfloor \frac{a \bmod 1461}{4} \right\rfloor + 2 \\ (y, m, d) &= \left(\left\lfloor \frac{a}{1461} \right\rfloor, \left\lfloor \frac{b}{153} \right\rfloor + 3, \left\lfloor \frac{b \bmod 153}{5} \right\rfloor + 1 \right) \end{aligned}$$

- 条文に対する論理式は日付けシステムとは独立になっているので，どのような日付けシステムを用いてもその記述には変化はないが，そこで使われる基本用語 (**属月**，**年齢**，**期間**，**月内で最後に成立**など) の定義は日付けシステムに依存するので当然変化する．これらは用語の定義集 `definitions` に纏められているが，修正ユリウス通日にもとづく日付けシステムを用いた場合の定義式は，`definitions_date` に与えられている (第 10 章参照)．

  通日/西暦の変換式では `floor` 関数が多用されているが，本書第 4 版ではその実現に変換式全体を `Exists` により量限定していたために変換式の実行に時間が掛かっていた．本版では，大井による `quot` 関数の導入により `Exists` を使わない表現が可能になり計算時間の短

縮が可能になった [4].

- 年金原簿も日付けシステムに依存する．第七条の検証で原理的検証で honnin_v7 では単純化されたものを用いたが，年金支給要件や年金額の計算を規定している第二十六条，第二十七条の検証には，西暦による日付けに対応した honnin_ladyA_date を用いた．

8.2 抽象日付けデータを用いた検証

国民年金法の条文の論理式記述に現れる述語は，基本的には日付に関するものであり，ある起点からの日数上の述語として扱うことが出来るので，日数を Z3 の組み込みデータ型である整数 Int 型で表現し，整数上の演算を用いて日付に関する演算を実現してきた．

一方，Z3Py では抽象データ型を定義し，それ上の変数に関する充足性判定を行うことが可能であり，この機能を使うことにより抽象的な検証を行える可能性がある．ここでは，日付に関する抽象データ型を定義し，第七条および第十八条に関する一般的な性質の検証事例を述べる．

8.2.1 抽象日付データを用いた第七条の検証

defintions_v7_abs では，数え上げ型 Day の抽象日 D1〜D6 と，それ上の全順序関係 p が定義されている．年金原簿 honnin_v7_abs では，被保険者の資格に現れる述語が，端点を抽象日とする区間として与えられている．検証 v7_abs は，このような一般的な状況の下でも 3 種類の被保険者の種別が共に成立する日がないことと，および，第三号被保険者になり得るために日データ上の順序関係 p が満たすべき条件が充足性判定の結果として求められている．充足性判定の対象は d と p である．

```
# defintions_v7_abs.py
from essentials import *

Day,(D1,D2,D3,D4,D5,D6)= \
        EnumSort('Day',('D1','D2','D3','D4','D5','D6'))
p=Function('p',Day,Day,BoolSort())
x,y,z=Consts('x y z',Day)
orderDay=And(
        ForAll(x,p(x,x)),
        ForAll([x,y,z],Implies(And(p(x,y),p(y,z)),p(x,z))),
        ForAll([x,y],Implies(And(p(x,y),p(y,x)),x==y)),
        ForAll([x,y],Or(p(x,y),p(y,x))))
between=(lambda x,z:lambda y:And(p(x,y),p(y,z)))
```

```
# honnin_v7_abs.py
# 第七条被保険者の検証で使用
from essentials import *
```

```
二十歳以上六十歳未満=(lambda d:True)
日本国内に住所を有する=(lambda d:between(D1,D2)(d))
厚生年金保険法老齢等受給可能=(lambda d:False)
厚生年金保険の被保険者=(lambda d:between(D3,D4)(d))
第二号被保険者の配偶者=(lambda d:between(D5,D6)(d))
主に第二号被保険者の収入により生計維持=(lambda d:between(D5,D6)(d))
```

```
# v7_abs.py
# 抽象的日付データに基づく検証
import dummy
dummy.honnin='honnin_v7_abs'
dummy.definitions='definitions_v7_abs'

from f7 import * # 第七条被保険者の資格
d=Const('d',Day)
s=Solver()
s.add(orderDay)
# 同時に異なる種別の被保険者にはなれないことの検証
s.push()
s.add(Or(And(第一号被保険者(d),第二号被保険者(d)),
         And(第一号被保険者(d),第三号被保険者(d)),
         And(第二号被保険者(d),第三号被保険者(d))))
print(s.check())
# unsat

# 第三号被保険者になる日が存在し得ることの検証
s.pop()
s.add(第三号被保険者(d))
print(s.check())
print(s.model())

# sat
# [d = D1,
#  p = [else ->
#       Or(And(Var(0) == D6, Var(1) == D2),
```

```
#           And(Var(0) == D4, Var(1) == D4),
#           And(Var(0) == D2, Var(1) == D3),
#           And(Var(0) == D2, Var(1) == D2),
#           And(Var(0) == D4, Var(1) == D5),
#           And(Var(0) == D5, Var(1) == D5),
#           And(Var(0) == D3, Var(1) == D3),
#           And(Var(0) == D1, Var(1) == D2),
#           And(Var(0) == D4, Var(1) == D3),
#           And(Var(0) == D6, Var(1) == D6),
#           And(Var(0) == D1, Var(1) == D3),
#           And(Var(0) == D1, Var(1) == D6),
#           And(Var(0) == D6, Var(1) == D3),
#           And(Var(0) == D5, Var(1) == D1),
#           And(Var(0) == D1, Var(1) == D1),
#           And(Var(0) == D4, Var(1) == D6),
#           And(Var(0) == D5, Var(1) == D2),
#           And(Var(0) == D4, Var(1) == D1),
#           And(Var(0) == D4, Var(1) == D2),
#           And(Var(0) == D5, Var(1) == D6),
#           And(Var(0) == D5, Var(1) == D3))]]
# 抽象日データ上の全順序関係 p は，p(x,y),x!=y を x<y と書くと，このモデルは
#         D4 < D5 < D1 < D6 < D2 < D3
# となっており，d=D1 とすると，
#         D5 < d < D6
# となり，第二号被保険者の配偶者の期間を満たすが，
#         D4 < d < D3
# であるので厚生年金保険の被保険者の期間を満たしておらず，
# 確かに日 d=D1 では，第三号被保険者であることがわかる．
# このようなモデルは唯一でない．
```

8.2.2 抽象日付データを用いた第十八条の検証

第十八条年金の支給期間及び支払期月の抽象日付けデータによる検証には，日と月の抽象化とともに日と月の関係の抽象化が必要になる（**支給事由発生**は日に関する述語であり，**受給権月**は月に関する述語である）．これに伴い，月と日の関連する**属月**，**期間**などの関数や，その定義に現れる m+1 も**次月**として抽象化を行う必要がある．definitions_v18_abs では，これらを含む日付けに関する抽象化が与えられている．

検証は，（１）支給事由が発生した日の属月の次月から停止事由が成立する月まで支給が行われること (prop1)，（２）停止事由が発生した月の次月から停止事由の成立する月まで支給は停止されること (prop2)，（３）停止事由が消滅した月の次月から受給権の消滅する月まで支給が行われること (prop3)，（４）支給と停止が同時に成立する月がないこと (prop4) について行った．

検証のキーポイントは，抽象的な日付の構造をどのように定めるかであり，definitions_v18_abs に与えられている．以下では，これについて説明する．

1. 事象の発生する日:
   支給事由発生日:D1，**停止事由発生日**:D2，**停止事由消滅日**:D3，**受給権利消滅日**:D4
   これらは抽象データとして数え上げ型 Day を構成し，この上で全順序関係 p が成り立ち，それに関しこれらの日は D1,D2,D3,D4 の順に並んでいる (seqDay).
2. 関係する月:
   条文記述に現れ，また，検証に関連する月は以下である．

   ```
   M1 M11 .. M112 .. M2 M22 .. M223 .. M3 M33 .. M334 .. M4
   D1                 D2                 D3                 D4
   ```

   月データも数え上げ型 Month として定義され，全順序関係 q が定義され，各 M は上の順番に並んでいる (seqMonth).
   - Mi は日 Di の属する月（**属月**の定義参照）(i=1,2,3,4)
   - Mii は月 Mi の次の月（**次月**の定義参照）
   - Miij は月 Mii と月 Mj の間の任意の月

   なお，月 MM は，**属月**，**次月**の定義に使われた If 関数の else 部のパディングであり，実際に参照されることはない．
3. 述語**期間**は，第十八条において導入されたものを，以上の定義の下に抽象日付 Day,Month 上に定義し直したものである．

prop1,2,3,4 の検証結果から，支給事由の発生から権利の消滅までの間に支給停止期間があるという設定に関しては，論理式表現 f18 が正しいことが一般的な条件の下で確認された．なお，prop1,2,3 に含まれる関数変数 p,q に対しては seqDay,seqMonth によりその解釈が一意に定まることから（正確には，MM の自由度を除いて），検証結果が sat であることはそれらが正しいことを表している．

```
# f18.py　第十八条　年金の支給期間及び支払期月，通則版
from essentials import *

受給権月=期間（支給事由発生，受給権利消滅）
支給=(lambda m:And(Not(停止（m)），受給権月（m)))
停止=期間（停止事由発生，停止事由消滅）
年金支払額=(lambda m:
            If(偶数月（m)，年金支給額（m-2)+ 年金支給額（m-1),0))
年金支給額=(lambda m:If(支給（m)，年金額（m),0))
```

```
# definitions_v18_abs.py
from essentials import *

Day,(D1,D2,D3,D4)=EnumSort('Day',('D1','D2','D3','D4'))
p=Function('p',Day,Day,BoolSort())
x,y,z=Consts('x y z',Day)
orderDay=And(
        ForAll(x,p(x,x)),
        ForAll([x,y,z],Implies(And(p(x,y),p(y,z)),p(x,z))),
        ForAll([x,y],Implies(And(p(x,y),p(y,x)),x==y)),
        ForAll([x,y],Or(p(x,y),p(y,x))))
seqDay=And(p(D1,D2),p(D2,D3),p(D3,D4))

Month,(M1,M2,M3,M4,M11,M22,M33,M112,M223,M334,MM)= \
        EnumSort('Month',('M1','M2','M3','M4','M11','M22','M33', \
                          'M112','M223','M334','MM'))
q=Function('q',Month,Month,BoolSort())
u,v,w=Consts('u v w',Month)
orderMonth=And(
        ForAll(u,q(u,u)),
        ForAll([u,v,w],Implies(And(q(u,v),q(v,w)),q(u,w))),
        ForAll([u,v],Implies(And(q(u,v),q(v,u)),u==v)),
        ForAll([u,v],Or(q(u,v),q(v,u))))
seqMonth=And(q(M1,M11),q(M11,M112),q(M112,M2),q(M2,M22),q(M22,M223),
             q(M223,M3),q(M3,M33),q(M33,M334),q(M334,M4))
```

```
属月=(lambda d:If(d==D1,M1,If(d==D2,M2,If(d==D3,M3,If(d==D4,M4,MM)))))
次月=(lambda m:If(m==M1,M11,If(m==M2,M22,If(m==M3,M33,MM))))
期間=(lambda f,g:(lambda m:
        Exists(x,
            And(f(x),q(次月 (属月 (x)),m),
                ForAll(y,
                    Implies(And(g(y),p(x,y)),
                            q(m, 属月 (y))))))))
```

```
#####################
# honnin_v18_abs.py #
#####################

# 第十八条の検証で使用

from essentials import *

支給事由発生=(lambda d:d==D1)
受給権利消滅=(lambda d:d==D4)
停止事由発生=(lambda d:d==D2)
停止事由消滅=(lambda d:d==D3)
```

```
# v18_abs.py
import dummy
dummy.definitions='definitions_v18_abs'

#from essentials import *
from f18 import *
d1,d2=Consts('d1 d2',Day)
m,m1,m2=Consts('m m1 m2',Month)
s=Solver()
s.add(orderMonth,orderDay,seqMonth,seqDay)
s.push()

# 支給事由が発生した月は支給されず，次月から支給が開始され，停止事由が成立する月まで
```

```
# 支給される.
prop1=ForAll([d1,d2,m1,m2,m],
        Implies(And(支給事由発生 (d1), 停止事由発生 (d2),m1==属月 (d1),m2==属月 (d2)),
                And(Not(支給 (m1)),
                    Implies(And(q(次月 (m1),m),q(m,m2)),
                            And(支給 (m))))))
s.add(prop1)
print(s.check())
# sat
print(s.model())
# [p = [else ->
#       Or(And(Var(0) == D4, Var(1) == D4),
#          And(Var(0) == D3, Var(1) == D3),
#          And(Var(0) == D1, Var(1) == D1),
#          And(Var(0) == D2, Var(1) == D2),
#          And(Var(0) == D3, Var(1) == D4),
#          And(Var(0) == D1, Var(1) == D3),
#          And(Var(0) == D1, Var(1) == D2),
#          And(Var(0) == D2, Var(1) == D3),
#          And(Var(0) == D1, Var(1) == D4),
#          And(Var(0) == D2, Var(1) == D4))],
#  q = [else ->
#       Or(And(Var(0) == M3, Var(1) == M3),
#          And(Var(0) == M2, Var(1) == M2),
#          And(Var(0) == M1, Var(1) == M3),
#          And(Var(0) == M223, Var(1) == M223),
#          And(Var(0) == M11, Var(1) == M4),
#          And(Var(0) == M112, Var(1) == M4),
#          And(Var(0) == M2, Var(1) == M33),
#          And(Var(0) == M2, Var(1) == M22),
#          And(Var(0) == M1, Var(1) == M4),
#          And(Var(0) == M22, Var(1) == M22),
#          And(Var(0) == M3, Var(1) == M33),
#          And(Var(0) == M11, Var(1) == M334),
#          And(Var(0) == M334, Var(1) == M334),
#          And(Var(0) == M223, Var(1) == M33),
#          And(Var(0) == M11, Var(1) == M11),
```

```
#           And(Var(0) == M22, Var(1) == M334),
#           And(Var(0) == M223, Var(1) == M4),
#           And(Var(0) == M112, Var(1) == M223),
#           And(Var(0) == M1, Var(1) == M33),
#           And(Var(0) == M11, Var(1) == M22),
#           And(Var(0) == M112, Var(1) == M112),
#           And(Var(0) == M1, Var(1) == M1),
#           And(Var(0) == M223, Var(1) == M334),
#           And(Var(0) == M22, Var(1) == M33),
#           And(Var(0) == M22, Var(1) == M4),
#           And(Var(0) == M2, Var(1) == M223),
#           And(Var(0) == M334, Var(1) == M4),
#           And(Var(0) == M1, Var(1) == M2),
#           And(Var(0) == M11, Var(1) == M33),
#           And(Var(0) == M22, Var(1) == MM),
#           And(Var(0) == M33, Var(1) == M334),
#           And(Var(0) == M3, Var(1) == M334),
#           And(Var(0) == M112, Var(1) == M3),
#           And(Var(0) == M1, Var(1) == M11),
#           And(Var(0) == M2, Var(1) == M3),
#           And(Var(0) == M112, Var(1) == M22),
#           And(Var(0) == M11, Var(1) == M112),
#           And(Var(0) == M33, Var(1) == M4),
#           And(Var(0) == M4, Var(1) == M4),
#           And(Var(0) == M2, Var(1) == M4),
#           And(Var(0) == M1, Var(1) == M22),
#           And(Var(0) == M112, Var(1) == M334),
#           And(Var(0) == M112, Var(1) == M2),
#           And(Var(0) == M33, Var(1) == M33),
#           And(Var(0) == M1, Var(1) == M112),
#           And(Var(0) == M1, Var(1) == M334),
#           And(Var(0) == M3, Var(1) == M4),
#           And(Var(0) == M223, Var(1) == M3),
#           And(Var(0) == M1, Var(1) == M223),
#           And(Var(0) == MM, Var(1) == M33),
#           And(Var(0) == M11, Var(1) == M223),
#           And(Var(0) == M2, Var(1) == MM),
```

```
#           And(Var(0) == M2, Var(1) == M334),
#           And(Var(0) == M11, Var(1) == M2),
#           And(Var(0) == MM, Var(1) == MM),
#           And(Var(0) == M112, Var(1) == M33),
#           And(Var(0) == M11, Var(1) == M3),
#           And(Var(0) == M22, Var(1) == M3),
#           And(Var(0) == M22, Var(1) == M223),
#           And(Var(0) == MM, Var(1) == M3),
#           And(Var(0) == MM, Var(1) == M334),
#           And(Var(0) == M1, Var(1) == MM),
#           And(Var(0) == MM, Var(1) == M223),
#           And(Var(0) == M112, Var(1) == MM),
#           And(Var(0) == M11, Var(1) == MM),
#           And(Var(0) == MM, Var(1) == M4))]]

s.pop()
s.push()

# 停止事由が発生した月の次月から停止事由が消滅する月まで支給は停止する.
prop2=ForAll([d1,d2,m1,m2,m],
        Implies(And(停止事由発生 (d1), 停止事由消滅 (d2),m1==属月 (d1),m2==属月 (d2)),
                And(Not(停止 (m1)),
                    Implies(And(q(次月 (m1),m),q(m,m2)),
                            And(停止 (m))))))
s.add(prop2)
print(s.check())
# sat

s.pop()
s.push()

# 停止事由が成立する月の次月から受給権が消滅する月まで支給される.
prop3=ForAll([d1,d2,m1,m2,m],
        Implies(And(停止事由消滅 (d1), 受給権利消滅 (d2),m1==属月 (d1),m2==属月 (d2)),
                And(Not(支給 (m1)),
                    Implies(And(q(次月 (m1),m),q(m,m2)),
                            And(支給 (m))))))
```

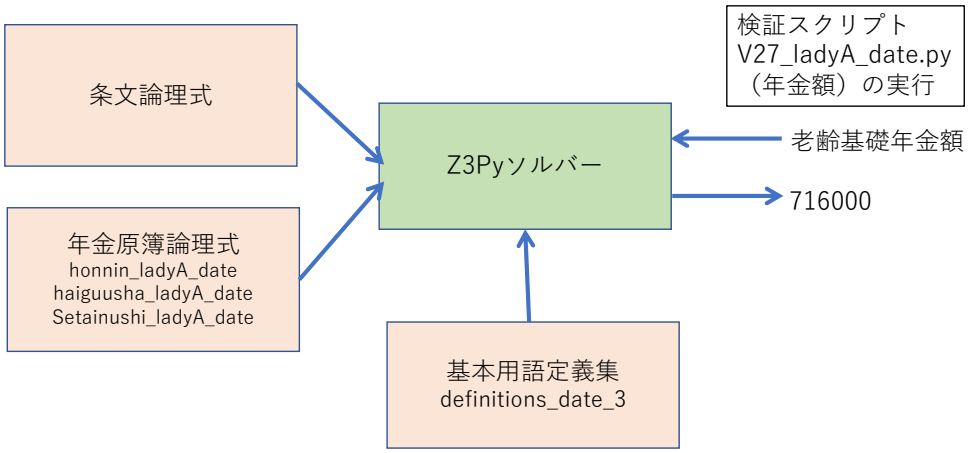
```
s.add(prop3)
print(s.check())
# sat

s.pop()
s.push()

# 支給と停止が共に成立する月はない.
prop4=And(支給 (m), 停止 (m))
s.add(prop4)
print(s.check())
# unsat
```

9　論理式記述に基づく年金システムシミュレーターの構成

- Z3Py などの SMT ソルバーを用いて条文論理式に対して充足性判定をすることにより，法令の「内容」を実行することが出来ることをこれまで見てきた．したがって，「法令論理式の集合＋ SMT ソルバー」を法令のシミュレーターと考えることができる．あるいは，これにユーザーインタフェースやデータベースを付加することにより法令実働化情報システムの基本アーキテクチャとすることが可能である．



- この方法で法令実働化システムを構築することは，条文論理式をそのままの形で動かす事になり，設計が容易で保守に強いシステム開発方法論になる．通常の法令実働化情報システム構築方法論では，システムの動作や操作を中心に分析が行われ，条文間のデータの受け渡しやそれを起こすための制御の流れを事前に解析してシステムの設計が行われる．一方，上に述べた方法では，このようなデータや制御の流れは SMT ソルバーの中で自動で行われるので，条文に対する論理式さえ作り上げれば基本的なシステム設計は終了する．
- これは，特に，法令の改正などに対して有利な方法論である．なぜなら，法令改正の多くは，少数の条文の変更に関して行われるのが普通だからであり，それら関連条文の論理式を変更することにより全体の再設計をする必要がないからである．
- この法令実働化システム構築法は，実行時の計算量を増やすことになるので実行速度の面では不利ではあるが，最近の計算機ハードウェアの高性能化や低コスト化により採用可能になったと考えることが出来る．それ以上に重要ことは，従来の構築法によると，度重なる法令改正に伴うシステム構造の劣化がその保守の困難性を大幅に引き上げ，運用を困難にする

可能性があることである．法令論理式の充足性判定に基づく方法は，この点に於いて極めて有利である．

- SMT ソルバーによる法令実働化システムの高速化は，ハードウェアの高性能化に求める以外にもアルゴリズム的側面からも考えられる．

  充足性判定により変数を含んだ論理式から変数値を自動で決定できることは条文論理式のデバッグにおいて非常に有効であるが，一旦正しい論理式が完成した後の（実運用時の）定数データに対しては，推論方式のこの柔軟性が実行速度の低下の主な原因となっていると考えられる．これに対し，論理式を関数的に解釈し推論方式を関数計算に切り替えることにより，定数データを対象とする推論を非常に高速に実行可能であることが判明している．詳細は別稿で述べる．

10　定義モジュール definitions

```
#########################
# definitions_simple.py #
#########################

# 論理式記述を行うために必要となる補助的な関数や述語の定義集
# 日付に関しては，1 月 30 日，1 年 12 月と簡略化して計算を高速化
# 一般的なカレンダーに関するものは，definitions_date に与えられている．

from essentials import *
d,d1,d2,d3=Ints('d d1 d2 d3')

# 老齢基礎年金改定率, 平成 30 年
改定率=0.998

#### 論理演算子の述語演算子化 ####
# And,Or,Not などの Bool 上の論理演算子を，通日に関する関数 AND,OR,NOT としたもの．
# 日付パラメータを使わずに，条文の言語表現に近い論理式化が可能となる．
# 内容的には，述語 p,q,..,r:Int=>Bool に対して，
#   AND(p,q,..,r)==lambda d: And(p(d),q(d),..,r(d))
#   OR(p,q,..,r)==lambda d: Or(p(d),q(d),..,r(d))
#   NOT(p)==lambda d: Not(p(d))
#   IF(p,q,r)==lambda d: If(p(d),q,r)
# d:Int は，通日を表す変数

from functools import reduce
OR=(lambda *args:(lambda d:
      reduce(Or,[x(d) for x in args],False)))
AND=(lambda *args:(lambda d:
      reduce(And,[x(d) for x in args],True)))
NOT=lambda f:(lambda d:Not(f(d)))
IF=lambda p,q,r:lambda d:If(p(d),q,r)
FALSE=lambda d:False
TRUE=lambda d:True
```

```
### 日付けのための述語，関数 ###
# 西暦 1 年 1 月 1 日を起点とした日数，月数で日，月を表現
# 日付けに関しては，1 年 12 月，1 月 30 日　と簡単化

属月=lambda d:d/30 # 通日 d の属する月
偶数月=lambda m:m%2==0

### 充足リスト ###
# f(i) を満たす最大 m 個の i からなるリスト

def 充足リスト (f,m):
    v=[]
    i,i0=Ints('i i0')
    s=Solver()
    p=f(i)
    s.add(p)
    for n in range(0,m):
        if s.check()==sat:
            i0=s.model()[i]
            v=v+[i0.as_long()]
            s.add(i!=i0)
        else:
            break
    return sorted(v)

# #### 月数　####
# 条件 f(i) を満たす i を最大 600 個までカウント（年金額を計算する f27 で使用．）
# 被保険者の資格に必要な 480 月（40 年間）をカバー
# i ごとに f(i) の成立を判定
# 月数 (f) は　一階論理式ではなく，Python プログラムにより計算される数値であり，
# したがって，例えば，月数 (f)==10　になるように f に含まれるパラメータを
# ソルバに自動で解かせることは出来ない．

def 月数 (f):
    m=600
    v=0
    i,i0=Ints('i i0')
```

```
    p=f(i)
    s=Solver()
    s.add(p)
    for n in range(0,m):
        if s.check()==sat:
            i0=s.model()[i]
            v=v+1
            s.add(i!=i0)
        else:
            break
    return v

####  期間  ####
# 以下の３つの「期間」マクロ（期間，期間２，期間３）と「月内で最後に成立」では，
# （１）一階論理式によるものと，（２）内部でソルバーを呼び出す Python 形式のものを
# 与えている．一階論理による定義は，実用的なサイズの問題には時間が掛かりすぎるが，
# 記号的検証が可能である．
#
# 一方，python 形式のものは，大きなサイズの問題でも動作するが，
# 内部でソルバーを呼び出して充足リストの計算をしているので，f,g に含まれる
# パラメータの調整をソルバーに行わせるなどの柔軟な処理は出来ない．
# 以下で，f,g,h:日->{True,False}

## 期間
# 条件 f を最初に満たす日 d1 の翌月から，条件 g を最初に満たす日 d2 の属月まで月 m の集合．
# （このような m に対して True となる述語）
# このような f,g の対が複数ある場合は，それらの期間を合算したもの．
# ...f.....g....f.....g...
#    -     -    =     =

# 一階論理式版
# 期間=(lambda f,g:(lambda m:
#     Exists(d1,
#         And(f(d1),属月(d1)+1<=m,
#             ForAll(d2,
#                Implies(And(g(d2),d1<=d2),
#                        m<=属月(d2)))))))
```

```
# Python 版
# f 成立=>g 成立時点を取り出し，その間の区間を Or で合算して期間を構成している.
# k は区間の最大数である.

def 期間 (f,g):
    def term(m):
        k=100
        p=False
        Lf=充足リスト (f,k)
        Lg=充足リスト (g,k)
        while Lf!=[]:
            m1=属月 (Lf[0])
            m2=属月 (Lg[0])
            p=Or(p,If(m1==m2,m==m1,And(m1<m,m<=m2)))
            Lf=Lf[1:]
            Lg=Lg[1:]
        return p
    return term

##  期間 2
# f が成立に至った日の属月から、g が成立に至った日の前月までの月からなる期間
# ただし，f の成立した後で同じ月に g が成立したときには，その月は含める.
# このようなペアが複数ある場合はそれらの期間の集合

# logic 版
#
# 期間 2=(lambda f,g:(lambda m:
#     Exists(d1,
#         And(f(d1),属月 (d1)<=m,
#             Or(ForAll(d2,
#                  Implies(And(g(d2),d1<=d2),
#                          m<=属月 (d2)-1)),
#                And(属月 (d1)==m,
#                    Exists(d2,And(g(d2),属月 (d2)==m)))))))

# Python 版
```

```
def 期間2(f,g):
    def term2(m):
        k=100
        p=False
        Lf=充足リスト(f,k)
        Lg=充足リスト(g,k)
        while Lf!=[]:
            m1=属月(Lf[0])
            m2=属月(Lg[0])
            p=Or(p,If(m1==m2,m==m1,And(m1<=m,m<m2)))
            Lf=Lf[1:]
            Lg=Lg[1:]
        return p
    return term2

##  期間3
# fが成立する日の前月から、fが成立しなくなる日の月までの月の集合

# logic版
# 期間3=(lambda f:(lambda m:
#     Exists(d1,
#         And(f(d1),属月(d1)-1<=m,
#             ForAll(d2,
#                Implies(And(Not(f(d2)),d1<=d2),
#                        m<=属月(d2)))))))

# Python版

def 期間3(f):
    h1=成立に至る(f)
    h2=成立に至る(NOT(f))
    def term3(m):
        k=100
        p=False
        Lh1=充足リスト(h1,k)
        Lh2=充足リスト(h2,k)
```

```
        while Lh1!=[]:
            m1=属月(Lh1[0])
            m2=属月(Lh2[0])
            p=Or(p,And(m1-1<=m,m<=m2))
            Lh1=Lh1[1:]
            Lh2=Lh2[1:]
        return p
    return term3

###  事由の成立関係  ###
# 月 m のうちで，条件 f が成立する日 d があり，その後 g,h が成立する日はない.

# logic 版
# 月内で最後に成立=(lambda f,g,h:(lambda m:
#     Exists(d,
#         And(m==属月(d),
#             f(d),
#             ForAll(d1,
#                 Implies(And(d<=d1,m==属月(d1)),
#                         Not(Or(g(d1),h(d1))))))))))

# Python 版
def 月内で最後に成立(f,g,h):
    f1=lambda d:And(Not(f(d-1)),f(d))
    f2=lambda d:And(f(d),Not(f(d+1)))
    g1=lambda d:And(Not(g(d-1)),g(d))
    h1=lambda d:And(Not(h(d-1)),h(d))
    def term4(m):
        k=100
        p=False
        Lf1=充足リスト(f1,k)
        Lf2=充足リスト(f2,k)
        Lg1=充足リスト(g1,k)
        Lh1=充足リスト(h1,k)
        while Lf1!=[]:
            m1=属月(Lf1[0])
            m2=属月(Lf2[0])
```

```
            for d in Lg1+Lh1:
                if Lf2[0]<d and m2==属月 (d):
                    m2=m2-1
                    break
            p=Or(p,And(m1<=m,m<=m2))
            Lf1=Lf1[1:]
            Lf2=Lf2[1:]
        return p
    return term4

#  事由の成立
成立に至る=lambda f:lambda d: And(Not(f(d-1)),f(d))
過去に成立 = lambda f: lambda d:Exists(d1,And(d1<=d,f(d1)))
初めて成立 = lambda f: lambda d:And(f(d),
                                   ForAll(d1,Implies(f(d1),d<=d1)))

#  事由成立の前後関係
before=lambda d:lambda g:Exists(d2,And(g(d2),d2<d))
after=lambda d:lambda g:Exists(d2,And(g(d2),d<d2))
between=(lambda d1,d3:lambda g:
        Exists(d2,And(g(d2),d1<=d2,d2<=d3)))
```

```
#########################
# definitions_date_3.py #
#########################

# 論理式記述を行うために必要となる補助的な関数や述語の定義集
# 西暦による日付表記に対応

from essentials import *
d,d1,d2,d3=Ints('d d1 d2 d3')
y,m,m1,m2,mjd=Ints('y m m1 m2 mjd')

#### 老齢基礎年金改定率, 平成 30 年 ####
改定率=0.998

#### 論理式記述に表れる日付，日付表示 ####
# 日付：ある起点からの日数，月数，年数を用いて表す.
# これらを，通日，通月，通年と呼ぶ.
# 起点となる日は基本的には任意であるが，修正ユリウス通日では 1858 年 11 月 17 日を用いている.
# 本書でもこれを用いる.
# 日付表記：西暦による．和暦に関しては，年号のみを扱う関数を用いる.

#### 論理演算子の述語演算子化 ####
# And,Or,Not などの Bool 上の論理演算子を，通日に関する関数 AND,OR,NOT としたもの.
# 日付パラメータを使わずに，条文の言語表現に近い論理式化が可能となる.
# 内容的には，述語 p,q,..,r:Int=>Bool に対して,
#   AND(p,q,..,r)==lambda d: And(p(d),q(d),..,r(d))
#   OR(p,q,..,r)==lambda d: Or(p(d),q(d),..,r(d))
#   NOT(p)==lambda d: Not(p(d))
#   IF(p,q,r)==lambda d: If(p(d),q,r)
# d:Int は，通日を表す変数

from functools import reduce

OR=(lambda *args:lambda d:
      reduce(Or,[x(d) for x in args],False))
AND=(lambda *args:lambda d:
      reduce(And,[x(d) for x in args],True))
```

```
NOT=lambda f:lambda d:Not(f(d))
IF=lambda p,q,r:lambda d:If(p(d),q,r)
FALSE=lambda d:False
TRUE=lambda d:True

#### 日付けに関する述語，関数 ####
# 以下，「通日」，「通月」は，関数名とデータ集合名に同じ名前を使っている．
# 年，月，日，通日，通月，誕生日，年齢：Int
# 通日：年 x 月 x 日 => 通日
# 通月：年 x 月  => 通月
# Year：通日 => 年
# Month：通日 => 月
# Day：通日 => 日
# YMD：通日  =>  年 x 月 x 日
# 属月：通日 => 通月
# 属年：属月 => 通年
# 年齢：誕生日 => 通日 => 年齢
# 年齢到達日：年齢 x 誕生日 => 通日
# 年齢到達：(年齢 x 誕生日)x 通日 => Bool
# 昭和，平成，令和：和暦年 => 西暦年

quot=lambda a,b:(a-a%b)/b
# z3 では，Int a,b のとき整数除算，Python 中の実行では実数除算に注意

# 以下，通日，Year，Month，Day の定義は，Wikipedia による．

def 通日 (y,m,d):
        y1=If(m<=2,y-1,y)
        m1=If(m<=2,m+12,m)
        d1=quot(36525*y1,100)
        d2=quot(y1,400)
        d3=quot(y1,100)
        m2=quot(3059*(m1-2),100)
        mjd=d1+d2-d3+m2+d-678912
        return mjd

def Year(mjd):
```

```
        n=mjd+678881
        a2=quot(4*(n+1),146097)+1
        a1=quot(3*a2,4)
        a=4*n+3+4*a1
        b1=quot((a%1461),4)
        b=5*b1+2
        y1=quot(a,1461)
        m1=quot(b,153)+3
        y=If(m1>=13,y1+1,y1)
        return y

def Month(mjd):
        n=mjd+678881
        a2=quot(4*(n+1),146097)+1
        a1=quot(3*a2,4)
        a=4*n+3+4*a1
        b1=quot((a%1461),4)
        b=5*b1+2
        m1=quot(b,153)+3
        m=If(m1>=13,m1-12,m1)
        return m

def Day(mjd):
        n=mjd+678881
        a2=quot(4*(n+1),146097)+1
        a1=quot(3*a2,4)
        a=4*n+3+4*a1
        b1=quot(a%1461,4)
        b=5*b1+2
        d=quot(b%153,5)+1
        return d

def YMD(mjd):
        s=Solver()
        s.add(Year(mjd)==y,Month(mjd)==m,Day(mjd)==d)
        s.check()
        y1=s.model()[y]
```

```
        m1=s.model()[m]
        d1=s.model()[d]
        return (y1,m1,d1)

mjd2ymd=YMD

通月=lambda y,m:(y-1858)*12+m-11

# Python の中でのみ有効
def mjm2ym(mjm): # 通月=>年 x 月
    m=(mjm+10)%12+1
    y=(mjm-m+11)//12+1858
    return (y,m)

# Z3 版，Python で使うと y:real として出力
def YM(mjm): # 通月=>年 x 月
    m=(mjm+10)%12+1
    y=quot(mjm-m+11,12)+1858
    return (y,m)

# 通日 d の属する通月
属月=lambda d:(Year(d)-1858)*12+Month(d)-11
# 通月 m の属する通年
属年=lambda m:quot((m+10),12)+1858

偶数月=lambda m:m%2==1  # m:通月，通月は 1858 年 11 月が起点
前月=lambda f:lambda m:Exists(d,And(f(d),属月(d)==m+1))
# f の成立する日の属する月の前月

# 和暦年 => 西暦年
昭和=lambda y:y+1925
平成=lambda y:y+1988
令和=lambda y:y+2018

#### 日の区間，月の区間，当日，当月####
# 以下，x,x1,x2:（年，月，日）or（年，月），d:日，m:月
```

```
区間=lambda u,v:lambda w:And(u<=w,w<=v)

DD=lambda x1,x2:区間(通日(x1[0],x1[1],x1[2]),通日(x2[0],x2[1],x2[2]))
# DD((1977,4,1),(1998,9,30))
#            ==lambda d:And(通日(1977,4,1)<=d,d<=通日(1996,9,30))

MM=lambda x1,x2:区間(通月(x1[0],x1[1]),通月(x2[0],x2[1]))
# MM((1955,5),(1977,3))==lambda m:And(通月(1955,5)=<m,m=<通月(1977,3))

D=lambda x:lambda d:d==通日(x[0],x[1],x[2])
M=lambda x:lambda m:m==通月(x[0],x[1])
# D((1977,4,1))==lambda d:d==通日(1977,4,1)
#       通日(1977,4.1)にのみTrueになる述語

#### 年齢関連 ####
# 年齢：誕生日がbirthdayである人の通日dにおける年齢age
# Int => Int => Int

def 年齢(birthday): # birthday:通日
    def 年齢1(d):
        y=Year(d)
        a=y-Year(birthday)
        mjd=通日(y,Month(birthday),Day(birthday))
        age=a-If(d<mjd-1,1,0)
        return age
    return 年齢1

# 年齢到達：誕生日がbirthdayの人がage歳になる日dにのみTrueになる述語
# 年齢到達日：その日を与える関数
# 民法の年齢に関する規定で，誕生日の前日に年齢が加算される.

年齢到達=(lambda age,birthday:lambda d:
        d==通日(Year(birthday)+age,Month(birthday),Day(birthday)-1))
年齢到達日=(lambda age,birthday:
        通日(Year(birthday)+age,Month(birthday),Day(birthday)-1))
年齢到達月=lambda age,birthday:属月(年齢到達日(age,birthday))
```

```
#### 充足リスト ####
# f(i) を満たす相異なる最大 m 個のソートされた i からなるリスト

def 充足リスト (f,m):
    v=[]
    i,i0=Ints('i i0')
    p=f(i)
    s=Solver()
    s.add(p)
    for n in range(0,m):
        if s.check()==sat:
            i0=s.model()[i]
            v=v+[i0.as_long()]
            s.add(i!=i0)
        else:
            break
    return sorted(v)

#### 月数  ####
# 条件 f(i) を満たす i を最大 600 個までカウント（年金額を計算する f27 などで使用）
# 被保険者の資格に必要な 480 月（40 年間）をカバー
# i ごとに f(i) の成立を判定
# 月数 (f) は 一階論理式ではなく，Python プログラムにより計算される数値であり，
# したがって，例えば，月数 (f)==10　になるように f に含まれるパラメータを
# ソルバーに自動で解かせることは出来ない.

def 月数 (f):
    m=600
    v=0
    i,i0=Ints('i i0')
    p=f(i)
    s=Solver()
    s.add(p)
    for n in range(0,m):
        if s.check()==sat:
            i0=s.model()[i]
            v=v+1
```

```
            s.add(i!=i0)
        else:
            break
    return v

# 一方，以下の「月数」は，600 月分の式 If(f(i),1,0) を + で結合したもの.
# 月数を得るには，最後に一気に充足性判定により数値を計算する.
# f が簡単なものでないとこの「If 加算形式」のサイズが大きくなりすぎ，充足性判定に時間が掛かりすぎる.
# しかし，サイズ M の小さい記号的検証では,f のパラメータの調整をこの方式により行うことが可能である.

# def 月数 (f):
#     M=600
#     m=0
#     for n in range(0,M):
#         m=m+If(f(n),1,0)
#     return m
#
# 月数 (f)=0+If(f(0),1,0)+If(f(1),1,0)+...+If(f(599),1,0)
# f=lambda d:d*d*d<1000
# v=Int('v')
# solve(月数 (f)==v)
# v=10

####  期間  ####
# 以下の３つの「期間」マクロ（期間，期間２，期間３）と「月内で最後に成立」では,
# （１）一階論理式によるものと，（２）内部でソルバーを呼び出す Python 形式のものを
# 与えている．一階論理による定義は，実用的なサイズの問題には時間が掛かりすぎるが,
# 記号的検証が可能である.
#
# 一方，python 形式のものは，大きなサイズの問題でも動作するが,
# 内部でソルバーを呼び出して充足リストの計算をしているので，f,g に含まれる
# パラメータの調整をソルバーに行わせるなどの柔軟な処理は出来ない.
# 以下で，f,g,h:日->{True,False}

## 期間
# 条件 f を最初に満たす日 d1 の翌月から，条件 g を最初に満たす日 d2 の属月まで月 m の集合
# （このような m に対して True となる述語）
```

```
# このような f,g の対が複数ある場合は，それらの期間を合算したもの．
# ...f.....g....f.....g...
#    -     -    =     =

# 一階論理式版
# 期間=(lambda f,g:(lambda m:
#     Exists(d1,
#         And(f(d1),属月(d1)+1<=m,
#             ForAll(d2,
#                 Implies(And(g(d2),d1<=d2),
#                         m<=属月(d2)))))))

# Python 版
# f 成立時点から g 成立時点までを取り出し，その間の区間を Or で合算して期間を構成している．
# k は区間の最大数である．

def 期間(f,g):
    def term(m):
        k=100
        p=False
        Lf=充足リスト(f,k)
        Lg=充足リスト(g,k)
        while Lf!=[]:
            m1=属月(Lf[0])
            m2=属月(Lg[0])
            p=Or(p,If(m1==m2,m==m1,And(m1<m,m<=m2)))
            Lf=Lf[1:]
            Lg=Lg[1:]
        return p
    return term

##  期間 2
# f が成立に至った日の属月から、g が成立に至った日の前月までの月からなる期間
# ただし，f の成立した後で同じ月に g が成立したときには，その月は含める．
# このようなペアが複数ある場合はそれらの期間の集合

# 一階論理式版
```

```
#
# 期間2=(lambda f,g:(lambda m:
#     Exists(d1,
#         And(f(d1),属月(d1)<=m,
#             Or(ForAll(d2,
#                  Implies(And(g(d2),d1<=d2),
#                          m<=属月(d2)-1)),
#                And(属月(d1)==m,
#                    Exists(d2,And(g(d2),属月(d2)==m)))))))

# Python版

def 期間2(f,g):
    def term2(m):
        k=100
        p=False
        Lf=充足リスト(f,k)
        Lg=充足リスト(g,k)
        while Lf!=[]:
            m1=属月(Lf[0])
            m2=属月(Lg[0])
            p=Or(p,If(m1==m2,m==m1,And(m1<=m,m<m2)))
            Lf=Lf[1:]
            Lg=Lg[1:]
        return p
    return term2

##  期間3
# fが成立する日の前月から、fが成立しなくなる日の月までの月の集合

# 一階論理式版
# 期間3=(lambda f:(lambda m:
#     Exists(d1,
#         And(f(d1),属月(d1)-1<=m,
#             ForAll(d2,
#                Implies(And(Not(f(d2)),d1<=d2),
#                        m<=属月(d2)))))))
```

```
# Python 版

def 期間 3(f):
    h1=成立に至る (f)
    h2=成立に至る (NOT(f))
    def term3(m):
        k=100
        p=False
        Lh1=充足リスト (h1,k)
        Lh2=充足リスト (h2,k)
        while Lh1!=[]:
            m1=属月 (Lh1[0])
            m2=属月 (Lh2[0])
            p=Or(p,And(m1-1<=m,m<=m2))
            Lh1=Lh1[1:]
            Lh2=Lh2[1:]
        return p
    return term3

####  事由の成立関係  ####

## 月内で最後に成立
# 月 m のうちで，条件 f が成立する日 d があり，その後 g,h が成立する日はない.

# logic 版
# 月内で最後に成立=(lambda f,g,h:(lambda m:
#     Exists(d,
#         And(m==属月 (d),
#             f(d),
#             ForAll(d1,
#                 Implies(And(d<=d1,m==属月 (d1)),
#                         Not(Or(g(d1),h(d1))))))))))

# Python 版
# 以下のものは，f が g,h と同時には成立しない，即ち,f(d) が g(d) や h(d) と共に成立する
# d が存在しない場合について，上記を最適化した Python プログラムであり，月 m の内で
```

```
# f が最後に成立する m を表す論理式を与える.

def 月内で最後に成立 (f,g,h):
    f1=lambda d:And(Not(f(d-1)),f(d))
    f2=lambda d:And(f(d),Not(f(d+1)))
    g1=lambda d:And(Not(g(d-1)),g(d))
    h1=lambda d:And(Not(h(d-1)),h(d))
    def term4(m):
        k=100
        p=False
        Lf1=充足リスト (f1,k)
        Lf2=充足リスト (f2,k)
        Lg1=充足リスト (g1,k)
        Lh1=充足リスト (h1,k)
        while Lf1!=[]:
            m1=属月 (Lf1[0])
            m2=属月 (Lf2[0])
            for d in Lg1+Lh1:
                if Lf2[0]<d and m2==属月 (d):
                    m2=m2-1
                    break
            p=Or(p,And(m1<=m,m<=m2))
            Lf1=Lf1[1:]
            Lf2=Lf2[1:]
        return p
    return term4

## 事由の成立

成立に至る=lambda f:lambda d: And(Not(f(d-1)),f(d))
過去に成立 = lambda f:lambda d:Exists(d1,And(d1<=d,f(d1)))
初めて成立 = lambda f:lambda d:And(f(d),
                                   ForAll(d1,Implies(f(d1),d<=d1)))

## 事由成立の前後関係
# before: 日 d より前に g が成立する日がある.
```

```
before=lambda d:lambda g:Exists(d2,And(g(d2),d2<d))
after=lambda d:lambda g:Exists(d2,And(g(d2),d<d2))
between=(lambda d1,d3:lambda g:
        Exists(d2,And(g(d2),d1<=d2,d2<=d3)))
```

11　年金原簿モジュール

```
################
# honnin_v5.py #
################

# 第五条の論理式化 f5 の正しさを検証するための小さな年金原簿
# f5 の検証には，この他に世帯主，配偶者に関する haiguusha_v5,setainushi_v5 が必要
# f11 の検証にも使用

from essentials import *

# 第七条関連
年齢=(lambda d:26+d/360)
二十歳以上六十歳未満=(lambda d:And(20<=年齢 (d), 年齢 (d)<60))
日本国内に住所を有する=(lambda d:And(200<d,d<600))
厚生年金保険法老齢等受給可能=(lambda d:False)
厚生年金保険の被保険者=(lambda d:And(300<d,d<500))
第二号被保険者の配偶者=(lambda d:And(400<d,d<550))
主に第二号被保険者の収入により生計維持=(lambda d:And(400<d,d<550))

# 保険料納付状況
保険料納付済=(lambda m: Or(m==6,m==9,m==19))
残りの四分の一納付済=(lambda m:False )
残りの半額納付済=(lambda m:Or(m==7,m==8))
残りの四分の三納付済=(lambda m:False)
追納により納付とみなされる期間=(lambda m:False)

# 保険料免除関係
保険料法定免除済=(lambda m: False)
保険料全額免除済=(lambda m: False)
保険料四分の三免除=(lambda m: False)
保険料半額免除=lambda m: Or(m==7,m==8)
保険料四分の一免除=(lambda m: False)
保険料免除_学生済=(lambda m:m==18)
学生等=(lambda m:m==18)
```

```
# 家族関係
世帯主が本人以外=(lambda m:False)
配偶者がいる=(lambda m:False)

#第八十九条関係
障害基礎年金の受給権者=(lambda d:False)
厚生年金保険法に基づく障害を支給事由とする年金給付の受給者=(lambda d:False)
障害を支給事由とする政令で定める給付の受給権者=(lambda d:False)
最後に障害状態に該当しなくなつた日から三年を経過=(lambda d: False)
障害状態にない=(lambda d:False)
政令で定める者=(lambda d:False) # 国民年金法施行令第六条の五第二項
生活保護法の生活扶助を受けている者=(lambda d:False)
厚生労働省令施設の入所者=(lambda d:False)
納付不要保険料納付申出=(lambda m:False)

#第九十条，九十条の二，九十条の三関係
前年の所得が政令第六条の七で定める額以下=(lambda m:False)
前年の所得が政令第六条の八で定める額以下=(lambda m:Or(m==7,m==8))
前年の所得が政令第六条の九で定める額以下=(lambda m:False)
生活保護以外の厚生労働省令で定める援助を受給=(lambda m:False)
障害者であり前年の所得が政令で定める額以下=(lambda m:False)
寡婦であり前年の所得が政令で定める額以下=(lambda m:False)
天災など省令により保険料納付が著しく困難=(lambda m:m==18)
```

```
# haiguusha_v5.py
# 配偶者の年金原簿　第九十条，九十条の二で参照
# f5 の検証 v5 において，honnin_v5,setainushi_v5 と共に使用

from essentials import *

前年の所得が政令第六条の七で定める額以下=(lambda m: False)
前年の所得が政令第六条の八で定める額以下=(lambda m: Or(m==7,m==8))
前年の所得が政令第六条の九で定める額以下=(lambda m: False)
生活保護以外の厚生労働省令で定める援助を受給=(lambda m: False)
障害者であり前年の所得が政令で定める額以下=(lambda m: False)
```

寡婦であり前年の所得が政令で定める額以下=(lambda m:False)
天災など省令により保険料納付が著しく困難=(lambda m:False)

```
# setainushi_v5.py
# 世帯主の年金原簿　第九十条，九十条の二で参照
# f5 の検証 v5 において，honnin_v5,haiguusha_v5 と共に使用
from essentials import *

前年の所得が政令第六条の七で定める額以下=(lambda m: m==8)
前年の所得が政令第六条の八で定める額以下=(lambda m: Or(m==7,m==8))
前年の所得が政令第六条の九で定める額以下=(lambda m: False)
生活保護以外の厚生労働省令で定める援助を受給=(lambda m: False)
障害者であり前年の所得が政令で定める額以下=(lambda m: False)
寡婦であり前年の所得が政令で定める額以下=(lambda m:False)
天災など省令により保険料納付が著しく困難=(lambda m:False)
```

```
################
# honnin_v7.py #
################

# この年金原簿は，第七条，第八条，第九条，第十一条の検証で使用される．
# 日 d は，26 才の誕生日を起点とする通日で表現．1 年 360 日，1 月 30 日と簡単化
# なお，「年齢」の扱いに関しては，一般的なものが definition_date_3 に与えられている．

from essentials import *

年齢=(lambda d:26+d/360)
二十歳以上六十歳未満=(lambda d:And(20<=年齢(d),年齢(d)<60))
日本国内に住所を有する=(lambda d:And(200<d,d<600))
厚生年金保険法老齢等受給可能=(lambda d:False)
厚生年金保険の被保険者=(lambda d:And(300<d,d<500))
第二号被保険者の配偶者=(lambda d:And(400<d,d<550))
主に第二号被保険者の収入により生計維持=(lambda d:And(500<=d,d<550))

#     *----日本在住-----------------------------------------------------*
#     |                                                                 |
#     |             *-----企業に勤務---------------*                    |
#     |             |                               |                   |
#     |             |              *----結婚-----------------*          |
#     |             |              |                |          |         |
# ---*-------------*--------------*--------------*----------*---------*------
# d=201           301            401             499         549       599
#     |             |                               |          |         |
#     *--第一号----**----------第二号-------------**-第三号---**-第一号--*
# m= 6            9 10                      15       16   17     18    19

# なお，以下は honnin_v7 を第十一条期間の検証で用いた場合の各被保険者期間を示す．

# 第一号被保険者期間 [6，7，8，9，18，19]
# 第二号被保険者期間 [10，11，12，13，14，15]
# 第三号被保険者期間 [16，17]
```

```
# honnin_v7_func.py
# v7_func で使用
from essentials import *

二十歳以上六十歳未満=Function('p',IntSort(),BoolSort())
日本国内に住所を有する=Function('f',IntSort(),BoolSort())
厚生年金保険法老齢等受給可能=Function('q',IntSort(),BoolSort())
厚生年金保険の被保険者=Function('g',IntSort(),BoolSort())
第二号被保険者の配偶者=Function('h',IntSort(),BoolSort())
主に第二号被保険者の収入により生計維持=Function('k',IntSort(),BoolSort())
```

```
# honnin_v7_abs.py
# 第七条被保険者の検証で使用
from essentials import *

二十歳以上六十歳未満=(lambda d:True)
日本国内に住所を有する=(lambda d:between(D1,D2)(d))
厚生年金保険法老齢等受給可能=(lambda d:False)
厚生年金保険の被保険者=(lambda d:between(D3,D4)(d))
第二号被保険者の配偶者=(lambda d:between(D5,D6)(d))
主に第二号被保険者の収入により生計維持=(lambda d:between(D5,D6)(d))
```

```
######################
# honnin_v18_abs.py #
######################

# 第十八条の検証で使用

from essentials import *

支給事由発生=(lambda d:d==D1)
受給権利消滅=(lambda d:d==D4)
停止事由発生=(lambda d:d==D2)
停止事由消滅=(lambda d:d==D3)
```

```
#################
# honnin_v28.py #
#################

# 第二十八条の検証で使用
# 1 年 2 日と単純化し，65 才の誕生日を起点とする通日により日 d を表現
# 年齢=lambda d:65+d/2

from essentials import *

誕生日=130
六十五才に達した日=年齢到達日（65, 誕生日）
六十六才に達した日=年齢到達日（66, 誕生日）
七十才に達した日=年齢到達日（70, 誕生日）

老齢基礎年金の受給権者=lambda d:d>=0
障害基礎年金の受給権者=lambda d:d>=6
遺族基礎年金の受給権者=lambda d:False
付加年金の受給権者=lambda d:False
老齢を支給事由とするものを除く厚生年金受給権者=lambda d:False
老齢基礎年金請求日=20
支給繰下げ申出日=12

# 上の条件に対する検証結果と条件の順序関係を図示したもの
# sat [dd = 6]
# 年齢   65         66         67         68         69         70           71
# d      0----1----2----3----4----5----6----7----8----9----10----11----12---
#        |                                |                                 |
#    老齢支給事由                      障害支給事由                      繰下げ申出
#                                         |
#                                         V
#                                   支給繰下げ申請みなし日

# 以下は，各条件の順序関係などを変化させた場合の結果を染ます.

# 障害基礎年金の受給権者=lambda d:False
# 支給繰下げ申出日=12
```

```
# sat [dd = 10]
# 年齢  65        66        67        68        69        70          71
# d     0----1----2----3----4----5----6----7----8----9----10----11----12---
#       |                                                  |          |
#   老齢支給事由                                            |          繰下げ申出
#                                                          V
#                                                  支給繰下げ申請みなし日

# 障害基礎年金の受給権者=lambda d:False
# 支給繰下げ申出日=7
# sat [dd = 7]
# 年齢  65        66        67        68        69        70          71
# d     0----1----2----3----4----5----6----7----8----9----10----11----12---
#       |                             |
#   老齢支給事由                       繰下げ申出
#                                     |
#                                     V
#                                   支給繰下げ申請みなし日

# 老齢基礎年金請求=lambda d:d==1
# 支給繰下げ申出=lambda d:d==12
# unsat
# 年齢  65        66        67        68        69        70          71
# d     0----1----2----3----4----5----6----7----8----9----10----11----12---
#       |    |                                                      |
#  老齢支給事由|                                                    繰下げ申出
#           老齢受給請求
#                                   支給繰下げ申請みなし日：なし
```

```
########################
# honnin_ladyA_date.py #
########################

# A 女史略歴
# (60 才まで，日付けは西暦による)
# 1955.5.10 誕生
# 1973.4.1-1977.3.31 大学生
#   1975.4-1977.3 学生特例全額保険料免除
# 1977.4.1-1998.9.30 会社勤務
# 1985.6.1-1998.11.20 結婚，第二号被保険者の配偶者
# 1998.11.21-2008.1.31  離婚，自営業
#   1998.11-1999.11      保険料未納
#   1999.12-2007.1       保険料納付
#   2007.2-2008.1        保険料半額免除，残りの半額納付済
# 2008.2.1-2013.1.31  会社勤務
# 2013.2.1-2015.5.9  結婚，第二号被保険者の配偶者，2015.5.9 で 60 才

from essentials import *

誕生日=通日 (1955,5,10)
二十歳以上六十歳未満=区間 (年齢到達日 (20, 誕生日), 年齢到達日 (60, 誕生日)-1)
六十五歳に達する日=年齢到達日 (65, 誕生日)

# 家族関係 (月単位で)
世帯主が本人以外= \
        OR(MM((1955,5),(1977,3)),
           MM((1985,6),(1998,11)),
           MM((2013,2),(2015,5)))
配偶者がいる= \
        OR(MM((1985,6),(1998,11)),
           MM((2013,2),(2015,5)))

# 第七条被保険者の資格関係

日本国内に住所を有する=DD((1955,5,10),(2020,12,31))
厚生年金保険法老齢等受給可能=FALSE
```

```
厚生年金保険の被保険者= \
        OR(DD((1977,4,1),(1998,9,30)),
           DD((2008,2,1),(2013,1,31)))
第二号被保険者の配偶者= \
        OR(DD((1985,6,1),(1998,11,20)),
           DD((2013,2,1),(2020,12,31)))
主に第二号被保険者の収入により生計維持= \
        AND(第二号被保険者の配偶者,NOT(厚生年金保険の被保険者))

#保険料納付関係

保険料納付済=MM((1999,12),(2007,1))
残りの四分の一納付済=FALSE
残りの半額納付済=MM((2007,2),(2008,1))
残りの四分の三納付済=FALSE
追納により納付とみなされる期間=FALSE

#老齢基礎年金関係

老齢基礎年金請求=D((2020,5,9))
付加年金の受給権者=FALSE
老齢を支給理由とする厚生年金受給者=FALSE
障害基礎年金の受給権者=FALSE
遺族基礎年金の受給権者=FALSE
支給繰下げ申出=FALSE
繰下げによる増額率=(lambda d:1) # d:繰下げ確定日, 政令により定まる

## 保険料免除関係

保険料法定免除=FALSE
保険料全額免除=FALSE
保険料四分の三免除=FALSE
保険料半額免除=MM((2007,2),(2008,1))
保険料四分の一免除=FALSE
保険料免除_学生=MM((1975,4),(1977,3))
学生等=MM((1973,4),(1977,3))
保険料免除=OR(保険料法定免除, 保険料全額免除, 保険料四分の三免除,
```

保険料半額免除，保険料半額免除，保険料免除_学生）

#第八十九条関係

障害基礎年金の受給権者=FALSE
厚生年金保険法に基づく障害を支給事由とする年金給付の受給者=FALSE
障害を支給事由とする政令で定める給付の受給権者=FALSE
最後に障害状態に該当しなくなつた日から三年を経過=FALSE
障害状態にない=FALSE
施行令第六条の五第二項で定める者=FALSE # 国民年金法施行令第六条の五第二項
生活保護法の生活扶助を受けている者=FALSE
厚生労働省令施設の入所者=FALSE
納付不要保険料納付申出=FALSE

#第九十条，九十条の二，九十条の三関係

前年の所得が政令第六条の七で定める額以下=FALSE
前年の所得が政令第六条の八で定める額以下=MM((2007,2),(2008,1))
前年の所得が政令第六条の九で定める額以下=FALSE
生活保護以外の厚生労働省令で定める援助を受給=FALSE
障害者であり前年の所得が政令で定める額以下=FALSE
寡婦であり前年の所得が政令で定める額以下=FALSE
天災など省令により保険料納付が著しく困難=MM((1975,4),(1977,3))

```
# haiguusha_ladyA_date.py
# 配偶者の年金原簿　第九十条，九十条の二で使用

from essentials import *
```

前年の所得が政令第六条の七で定める額以下=FALSE
前年の所得が政令第六条の八で定める額以下=MM((2007,2),(2008,1))
前年の所得が政令第六条の九で定める額以下=FALSE
生活保護以外の厚生労働省令で定める援助を受給=FALSE
障害者であり前年の所得が政令で定める額以下=FALSE
寡婦であり前年の所得が政令で定める額以下=FALSE
天災など省令により保険料納付が著しく困難=FALSE

```
# setainushi_date.py
# 世帯主の年金原簿　第九十条，九十条の二で使用
from essentials import *

前年の所得が政令第六条の七で定める額以下=M((2008,1))
前年の所得が政令第六条の八で定める額以下=MM((2007,2),(2008,1))
前年の所得が政令第六条の九で定める額以下=FALSE
生活保護以外の厚生労働省令で定める援助を受給=FALSE
障害者であり前年の所得が政令で定める額以下=FALSE
寡婦であり前年の所得が政令で定める額以下=FALSE
天災など省令により保険料納付が著しく困難=FALSE
```

12 検証スクリプトからの年金原簿モジュール，定義モジュールの設定

条文論理式化の検証では，各条文は複数の年金原簿による異なった観点からの検証が行われるの普通である．例えば，第七条の検証では以下の 4 つの年金原簿を使用した．

honnin_v7，honnin_v7_func，honnin_v7_abs，honnin_ladyA_date

同様に条文の記述で使用される基本用語の定義も利用される年金原簿や検証内容に応じて切り替える可能性がある．それらの設定を検証スクリプト一箇所から一括して指定するために，以下の例で示すメカニズムを用いた．因みに，条文論理式の表現は使用する年金原簿/definitions モジュールから独立している．

- モジュール dummy

```
# dummy モジュール
# 検証スクリプトで指定する年金原簿モジュールや定義モジュールための変数の用意
honnin='dummy'
honnin_post='dummy'
haiguusha='dummy'
setainushi='dummy'
definitions='dummy'
```

- モジュール essentials

```
# z3 モジュール，年金銀簿モジュールおよび definitons モジュールを import
from z3 import *
from dummy import honnin,honnin_post,haiguusha,setainushi,definitions
exec('from '+definitions+' import *')
exec('import '+honnin+' as 本人')
exec('import '+honnin_post+' as 本人_post')
exec('import '+haiguusha+' as 配偶者')
exec('import '+setainushi+' as 世帯主')
exec('from '+honnin+' import *')
```

- 条文論理式モジュール　f7.py

```
from essentials import *
第一号被保険者=lambda d: \
            And(二十歳以上六十歳未満(d),
```

```
                日本国内に住所を有する(d),
                Not(厚生年金保険法老齢等受給可能(d)),
                Not(第二号被保険者(d)),
                Not(第三号被保険者(d)))
    ...
```

- 検証スクリプト　v7.py

```
    import dummy
    # 検証で使用する年金原簿モジュール，definitions モジュールを設定
    dummy.honnin='honnin_v7'
    dummy.definitions='definitions_simple'
    from essentials import *
    from f7 import *
    d=Int('d')
    s=Solver()
    p=Or(And(第一号被保険者(d),第二号被保険者(d)),
         And(第一号被保険者(d),第三号被保険者(d)),
         And(第二号被保険者(d),第三号被保険者(d)))
    ...
```

付録 A　論理式化から見た条文

国民年金法は，国民年金基金及び国民年金基金連合会，附則を除くと 165 条からなるが，ここでは，各条文がその内容に関して論理式記述にもとづく検査や検証が有意義なものであるかについて概観した．結論的には，全 165 条は以下の 3 つのカテゴリー a,b,c に分類される．この分類は概略的なものではあるが，広い意味で形式化が有効のものが全体の 7 割，その内で述語論理による形式化が有効なものが約 6 割あることがわかる．

a　条文中の用語や概念などに関して論理的関係が書かれており，論理式記述を行う事が条文の理解や分析に有効であるもの　73 条

b　複雑な論理的関係は書かれいないが，組織間の関連や権限の委譲などの書かれており，その形式化が法令の構造的理解や分析に有効なもの　42 条

c　財政，罰則などが書かれ，論理式記述の必要性が低いもの　50 条

以下，各条文ごとにカテゴリーを示す．（ただし，第十章国民年金基金及び国民年金基金連合会，附則は除く）

第一章　総則

第一条～第四条の三　c, 目的や財政など

第五条　用語の定義　a, 保険料の納付や免除に関する定義

第六条　事務の区分　b, 市町村における受託事務

第二章　被保険者

第七条　被保険者資格　a, 被保険者の資格の定義

第八条　資格取得の時期　a, 条文は不完全，また，その意味は薄い

第九条　資格喪失の時期　a, 同上

第十条　削除

第十一条, 十一条の二　被保険者期間の計算　a, 被保険者期間の計算に必要な定義

第十二条, 十二条の二　届け出　b, 被保険者の資格の変更などの届け出に関連して，被保険者や事業主，自治体，厚労大臣など関連組織の役割と関係

第十三条　年金手帳　b, 年金手帳の作成と交付　厚労大臣 ⇒ 被保険者

第十四条　年金原簿　a, 年金業務に必要な被保険者データの記録

第十四条の二～四 訂正の請求，方針，措置　b, 被保険者による年金原簿訂正請求 ⇒ 厚労大臣，社会保障審議会

第十四条の五　被保険者に関する情報の提供　b, 厚労大臣 ⇒ 被保険者

第三章　給付

第一節　通則

第十五条　給付の種類　a, 4 種類の給付，単純だがこの条文の形式化の仕方は基本的

第十六条　裁定　b,　厚労大臣、被保険者
第十六条の二　調整期間　　c, 政府が年金当別会計積立金などを勘案して給付額の調整
第十七条　端数処理　a, 50 銭四捨五入
第十八条　年金の支給期間及び支給期月　a, 年金の支給，停止を定める基本論理
第十八条の二　二月期支給の年金加算　a, 切り捨て端数の二月期への加算
第十八条の三　死亡の推定　a, 死亡に時期が明確で内場合の死亡推定の論理
第十八条の四　失踪　a, 失踪に関して同上
第十九条　未支給年金　a, 受給権者死亡の場合に年金を実際に受け取る人間の指定
第二十条　併給の調整　a, 複数の基礎年金が受給可能となった場合の選択方式
第二十条の二　受給権者の申出による支給停止　a
第二十一条、二十一条の二　年金の支払の調整　a, 誤って支払った年金は他の年金給付の内金
第二十二条　障害賠償請求権　b, 第三者が原因の傷害に関して，政府が受給権者から請求権を取得
第二十三条　不正利得の徴収　b, 偽りによる不正受給受給者 ⇒ 厚労大臣
第二十四条　受給権の保護　c, 年金権は担保，譲渡，差し押さえ不可
第二十五条　公課の禁止　c, 年金額を税金の算定額にできない
　第二節　老齢基礎年金
第二十六条　支給要件　a
第二十七条　年金額　a
第二十七の二、三、四、五　改定率の改定等　a
第二十八条　支給の繰下げ　a
第二十九条　失権　a
　第三節　障害基礎年金
第三十条、三十条の二、三、四　支給要件　a, 初診日，障害の程度，年齢などとの関係
第三十一条、三十二条　併給の調整　a, 新たな障害による受給権が発生した場合の調整
第三十三条、三十三条の二　年金額　a
第三十四条　障害の程度が変わつた場合の年金額の改定　a
第三十五条　失権　a
第三十六条、三十六条の二，三，四　支給停止　a
　第四節　遺族基礎年金
第三十七条　支給要件　a, 被保険者の子または配偶者に支給
第三十七の二　遺族の範囲　a
第三十八条　遺族の範囲　a
第三十九条、三十九条の二　年金額　a
第四十条 失権　a
第四十一条、四十一条の一、四十二条　支給停止　a
　第五節　付加年金、寡婦年金及び死亡一時金
　　第一款　付加年金

第四十三条　支給要件　a, 付加的保険料納付者が老齢基礎年金受給時
第四十四条　年金額　a
第四十五条　国民年金基金又は国民年金基金連合会の解散の場合の取扱い　c
第四十六条　支給の繰下げ　a, 老齢基礎年金の繰下と同期
第四十七条　支給停止　a, 老齢基礎年金と同期
第四十八条　失権　a, 同上
第二款　寡婦年金
第四十九条　支給要件　a, 夫が年金を貰う前に死亡した時に妻が受給
第五十条　年金額　a, 夫が貰う額の 3/4
第五十一条　失権　a
第五十二条　支給停止　a
第三款　死亡一時金
第五十二条の二　支給要件　a, 年金の受給まえに死亡した人の遺族に
第五十二条の三　遺族の範囲及び順位等　a
第五十二条の四、五　金額　a
第五十二条の六　支給の調整　a,　一時金　or 寡婦年金
第五十三条～六十八条　削除
第六節　給付の制限
第六十九条～七十三条　c, 故意の事故、書類の不提出、犯罪行為などの場合は支給しない。
第四章　国民年金事業の円滑な実施を図るための措置
第七十四条　c,　教育、広報、年金機構など
第五章　積立金の運用
第七十五条～八十条　c,　目的、運用、職員の責務、秘密保持義務、処分
第八十一条～八十四条　削除
第六章　費用
第八十五条　国庫負担　c, 毎年の国庫負担額を規定する数式
第八十六条　事務費の交付　c,　地方自治体に
第八十七条　保険料　a, 保険料の月額
第八十七条の二　a, 保険料の追加 400 円の納付, 付加年金のため
第八十八条　保険料の納付義務　b, 配偶者、世帯主、被保険者
第八十八条の二　a, 妊婦は免除
第八十九条　a, 法定免除
第九十条、九十条の二、三　a,　申請免除、学生特例
第九十一条　保険料の納期限　b,　被保険者 ⇒ 厚労大臣？
第九十二条　保険料の通知及び納付　b, 厚労大臣 ⇒ 被保険者（毎年度）
第九十二条の二　口座振替による納付　b,　通知：被保険者 ⇒ 厚労大臣
第九十二条の二の二　指定代理納付者による納付　b,　通知：被保険者 ⇒ 厚労大臣

第九十二条の三〜六　保険料の納付委託　　b,　被保険者、厚労大臣、納付受託者 (年金基金，市町村、...)、帳簿の立ち入り検査

第九十三条　保険料の前納　a

第九十四条　保険料の追納　a,　免除された保険料の納付

第九十四条の二、三、四　基礎年金拠出金　　c,　共済年金、厚生年金などから基礎年金に一定額の拠出

第九十四条の五　報告　　c,　厚労大臣、実施期間（共済組合など）

第九十四条の六　第二号被保険者及び第三号被保険者に係る特例　c, 第二、三号被保険者は保険料納付不要

第九十五条　徴収　c,　国税に倣って徴収

第九十五条の二　国民年金基金又は国民年金基金連合会の解散に伴う責任準備金相当額の徴収　c

第九十六条　督促及ぶ滞納処分　c, 具体的な督促、処分の方法

第九十七条　延滞金　c, 滞納者に対する延滞金の計算方法

第九十八条　先取特権　c, 徴収金の先取り特権の順位　国税、地方税の次

第九十九、百条　　削除

第七章　不服申し立て

第百一条　不服申し立て　c, 被保険者 ⇒ 社会保険審議会

第百一条の二　審査請求と訴訟との関係　c

第八章　雑則

第百二条　時効　c, 給付を受ける権利は 5 年，保険料の請求は 2 年

第百三条　期間の計算　c,　この法律内で定義されていなものは、民法による

第百四条　戸籍事項の無料証明　c

第百五条　届出等　b, 被保険者、受給権者、厚労大臣、市町村長

第百六条　被保険者に関する調査　b, 厚労大臣 ⇒ 被保険者

第百七条 受給権者に関する調査　b,　厚労大臣 ⇒ 受給権者

第百八条、第百八条の二、二の二，三　資料の提供など　b, 厚労大臣 ⇒ 関連諸機関

第百八条の四　基礎年金番号の利用制限等　b, 住民基本台帳に準ずる

第百九条　国民年金事務組合　b,　被保険者の行う届出 (第十二条）を行ってくれる事務組織

第百九条の二　全額免除申請の事務手続に関する特例　b,　被保険者に代わって申請手続き行うもの

第百九条の二の二　学生納付特例の事務手続に関する特例　b, 被保険者に代わって申請手続き行うもの

第百九条の三　保険料納付確認団体　b, 団体の被保険者の保険料が納付されていないことを確認 ⇒ 被保険者

第百九条の四　年金機構への厚生労働大臣の権限に係る事務の委任　　b, 厚労大臣 ⇒ 機構

第百九条の五　財務大臣への権限の委譲　b, 保険料未納者の対応を税務署に行わせるために、関連の権限を財務大臣に委譲

第百九条の六　機構が行う滞納処分等に係る認可等　b, 厚労大臣 ⇒ 機構

第百九条の七　滞納処分等実施規程の認可等　b, 厚労大臣 ⇒ 機構

第百九条の八　機構が行う立入検査等に係る認可等　b, 厚労大臣 ⇒ 機構

第百九条の九　地方厚生局長等への権限の委任　b, 厚労大臣 ⇒ 地方厚生局長等

第百九条の十　機構への事務の委託　b, 厚労大臣 ⇒ 機構

第百九条の十一　機構が行う収納　　b,　厚労大臣 ⇒ 機構, 保険料の徴収、日本銀行への送付

第百九条の十二　情報の提供　b,　機構 ⇒ 厚労大臣

第百九条の十三　厚生労働大臣と機構の密接な連携　b

第百九条の十四、十五条　研修、政令による経過措置　c

第百十条　実施命令　c, 詳細は省令で定める

第九章　罰則

第百十一〜百十四条　c

参考文献

[1] 国民年金法逐条解釈 (平成 27 年 4 月 1 日現在施行法令準拠), 厚生労働省 http://www.mhlw.go.jp/file/06-Seisakujouhou-12500000- Nenkinkyoku /0000095039.pdf

[2] 高木隆司：超解 年金法, 日本法令

[3] https://ja.wikipedia.org/wiki/ユリウス通日

[4] 大井理生：personal communication

著者紹介

片山卓也
東京工業大学卒業 (1962 年)
東京工業大学名誉教授, 北陸先端科学技術大学院大学名誉教授
現在 中央大学研究開発機構 客員研究員機構教授
専門 ソフトウェア工学，法令工学
katayama@jaist.ac.jp