Vol.2015-SE-187 No.41
2015/3/13

SMT ソルバを活用した業務ルールの等価性検証手法

伊藤信治[†1]　宮崎邦彦[†1]　伊藤翔[†1]

ビジネスルールは、法改正や顧客要求の変更に伴い、頻繁に変化し続けている。ビジネスルールを変更した場合の確認の観点の１つとして、入力と出力の関係が変更前と変更後で維持されるべき部分が維持されていること（等価性）の確認がある。等価性を確認する手段としては、サンプリングテストの活用がある。しかし、サンプリングテストの場合、等価性を保証できるのは、テストした範囲に限定されるという課題がある。本稿では、網羅的に等価性を確認する手段として、SMT ソルバを活用した手法について提案する。また、提案方式を行政事務上のビジネスルールに適用した結果について報告する。

Method for Verifying Equivalency of Business Rules using SMT Solver

SHINJI ITOH[†1]　KUNIHIKO MIYAZAKI[†1]　SHO ITO[†1]

Business rules continue to change by law amendment and change of customer's requirements. There is a confirmation of equivalency between pre-change rules and post-change rules as a view of confirmations in case of a change of business rules. Sampling testing technologies enable us to confirm the equivalency. However, there is a problem that the sampling testing technologies cannot guarantee the equivalency of the scope which we don't test. In this paper, we propose method for confirming the equivalency exhaustively using SMT solver and present the result which we applied the method to business rules about administrative affairs.

1. はじめに

公共分野や金融分野などで利用する業務システムは、法令、会社規則、商品・サービスなどで規定されているビジネスルールと整合性を保つ必要がある。ビジネスルールは、一度規定したら終わりではなく、社会環境の変化、他社との差別化・優位性確保などにより、変化し続けている。ビジネスルールが変化すれば、ビジネスルールとの整合性維持のため、関連する業務システムの変更も、当然のことながら必須である。また、ビジネスルールの変更は頻繁に行われており、かつ、業務システムへの反映も短期間で実施することが求められている。

このような課題を解決する手段として、ビジネスルール管理システム（BRMS）が注目されている 1)。BRMS は、ビジネスルールの作成、管理、実行などの機能を持ち、ビジネスルールの変更をコーディングレスでシステムに反映することができる。これにより、ビジネスルールの変更に伴うシステム改修を短期間で実施できる。

BRMS 製品の中には、ビジネスルール間の矛盾や漏れの有無を検証する機能などの検証機能を備えるものがある 2)3)。検証機能により、ビジネスルールの不具合を自動検出でき、システムの高品質化ができる。これらの既存の検証機能は、最新の（１バージョンの）ビジネスルールを対象にしたものである。

また、ビジネスルールを変更した場合には、不必要な変更が加えられていないこと、言い換えると、入力と出力の関係が変更前と変更後で維持されるべき部分が維持されていること（以下、「等価性」という。）を確認する必要がある。等価性を確認する手段としては、プログラムのテストで行うサンプリングテストがある。変更前のテストケースを活用することで、等価性を確認できる。しかし、サンプリングテストの場合、等価性を保証できるのは、テストした範囲に限定される。テストしていない範囲については、等価性を保証できないという課題がある。

このような背景の中で、著者らは、ビジネスルールの高品質化を目的とした検証技術の研究を進めている。これまでに、決定表を活用した矛盾や漏れを検証する技術の研究を実施した 4)。これらは、１バージョンのビジネスルールの検証技術である。

本稿では、変更前と変更後の２つのバージョンのビジネスルールに着目し、これらの等価性を検証する技術について検討した結果、および、提案方式を行政事務上のビジネスルールに適用した結果について報告する。

2. ビジネスルールの概要と仕様変更時の課題

2.1 ビジネスルールとは

ビジネスルールは、ビジネスを運用していく上でのガイドや判断基準である。例えば、法令に基づく給付金の支給判定や支給額計算、銀行における優遇金利の判定基準などがビジネスルールである。ビジネスルールの一部は、ソフトウェアに組み込まれ、その判断や計算などが自動化され

†1 (株)日立製作所
Hitachi Ltd.

ている。

ビジネスルールは、その性質に応じて、いくつかに分類できる。分類方法はさまざま提案されている。例えば、ビジネスルールに基づいた開発手法のパイオニアの１人である Barbara von Halle は、「用語」「事実」「制約ルール」「ガイドルール」「アクションルール」「計算ルール」「推論ルール」の７つに分類している 5)。

本研究では、if-then 形式で表現できるビジネスルールを対象とする。具体的には、上記７つの分類の内、制約ルール、ガイドルール、アクションルール、推論ルールである。用語は、ソフトウェア実装上データモデルとして表現でき、事実は備えるべき機能であり、ともに、一般に if-then 形式として実装されないため対象外とする。また、計算ルールは、一般に四則演算を活用して表現される数式であり、if-then 形式では表さないため対象外とする。

なお、次節以降では、1 つの if-then 形式のビジネスルールを単に「ルール」といい、ルールの集まりを「ルールセット」という。

2.2 ルールの形式

本研究で扱うルールは、データ項目として入力項目と出力項目の２種類があり、これらは区別される。また、ルールには、仕様ルールと条件制約ルールの２種類のルールがある。

仕様ルールは、入力と出力の関係を定義するルールであり、以下のように表現できる。

$$\text{If } Cond_{\alpha}(\boldsymbol{x}) \text{ then } Res_{\alpha}(\boldsymbol{y}) \qquad (1)$$

ここで、$Cond_{\alpha}(\boldsymbol{x})$は仕様ルールの条件部を表す論理式、$Res_{\alpha}(\boldsymbol{y})$は結果部を表す論理式、$\boldsymbol{x}$は入力項目、$\boldsymbol{y}$は出力項目、$\alpha$は仕様ルールの識別情報である。ある入力の組合せが条件部$Cond_{\alpha}(\boldsymbol{x})$を真にする場合、出力項目$\boldsymbol{y}$は結果部$Res_{\alpha}(\boldsymbol{y})$を真にすることを表す。一般的に、ルールと言えば、仕様ルールのことである。Barbara の分類では、ガイドルール、アクションルール、推論ルールの全て、および、制約ルールのうち入力と出力の関係を規定したものが仕様ルールである。

一方、条件制約ルールは、入力項目間の関係を定義するルールであり、以下のように表現できる。

$$\text{If } CondX_{\beta}(\boldsymbol{x}) \text{ then } ResX_{\beta}(\boldsymbol{x}) \qquad (2)$$

ここで、$CondX_{\beta}(\boldsymbol{x})$は条件制約ルールの条件部を表す論理式、$ResX_{\beta}(\boldsymbol{x})$は結果部を表す論理式、$\boldsymbol{x}$は入力項目、$\beta$は条件制約ルールの識別情報である。ある入力$\boldsymbol{x}$が条件部$CondX_{\beta}(\boldsymbol{x})$を真にする場合、入力$\boldsymbol{x}$は結果部$ResX_{\beta}(\boldsymbol{x})$も真にすることを表す。Barbara の分類では、制約ルールのうち入力項目間の関係を定義したものが条件制約ルールである。条件制約ルールを利用することで、現実に成立し得る入力の組合せを定義できる。逆に言えば、現実に成立し得ない入力の組合せを除外できる。例えば、「性別が男、かつ、年齢が１８歳未満、かつ、配偶者あり」という日本人は存在しない。民法第 731 条の婚姻適齢の規定を満たさないためである。このようなケースを除外するためには、以下のように条件制約ルールを記述すればよい。

If 性別=男 ∧ 配偶者=あり then 年齢≧ 18

また、条件制約ルールの集まりである条件制約ルールセット$\mathbb{C}$は、下式で表現できる。

$$\mathbb{C} \stackrel{\text{def}}{=} \bigwedge_{j \in \mathrm{CS}} (CondX_i(\boldsymbol{x}) \rightarrow ResX_i(\boldsymbol{x})) \qquad (3)$$

ここで、CS は、条件制約ルールの識別情報の集合である。

なお、$Cond_{\alpha}$、Res_{α}、$CondX_{\beta}$、$ResX_{\beta}$は、それぞれ論理式であり、比較演算子、論理演算子、入力項目、出力項目、定数値などを組合せて表現する。ただし、Res_{α}で利用できる比較演算子は等号のみであり、論理演算子は論理積のみである。この条件を満たさない場合、Res_{α}を満たす出力$\boldsymbol{y}$を一意に決定できない（非決定的となる）場合がある。実際には、仕様ルールの検討段階では、Res_{α}に不等号や論理和などが含まれる場合もあるが、最終的にプログラムとして実装される仕様ルールには含まれないことから、本研究では対象外とした。

2.3 ルール変更時の課題

ルールは、最初に定義したら終わりではなく、法改正や顧客要求の変更などの仕様変更により、入力と出力の関係を変更する必要がある。また、プログラムと同様に、度重なる変更による可読性の低下から、入力と出力の関係を維持したままルールを変更する、プログラムにおけるリファクタリング相当の作業を行う場合がある。

いずれの場合においても、ルールを変更した場合、レビューを行い、正しくルールが変更されているか、不必要な変更が加えられていないかなどを確認する必要がある。レビュー観点は、さまざまあるが、例えば、以下のようなものがある。

(1) 正確性：法改正等の内容を正確に反映しているか。
(2) 利用可能性 : 条件部を満たす入力の組合せが存在するか。
(3) 整合性：ルール間に矛盾はないか。
(4) 完全性：条件の考慮漏れはないか。
(5) 等価性 : 変更前後で、入力と出力の関係が維持されるべき箇所が維持されているか。

上記(1)については、レビュアーが、法改正内容や顧客要求内容との突合せを１つ１つ行う必要がある。(2)については、条件部の充足可能性判定により容易に判定可能である。(3)(4)については、決定表を活用した方法 4)が提案されている。

本稿では、(5)の等価性の検証について着目し、その方式

について検討した結果を述べる。

3. ビジネスルールの等価性検証方式

3.1 等価性の定義

本研究における等価性とは、変更前の仕様ルールセットRSのうち、変更後においても入力と出力の関係が維持される仕様ルールセット(以下、「等価性維持仕様ルールセット」という。)$ERS \subseteq RS$の条件部を満たす入力のすべての組合せについて、変更後の仕様ルールセットRS'の条件部のいずれかを満たし、かつ、条件部を満たす変更前と変更後の仕様ルールの結果部を満たす出力の組合せが存在することである。すなわち、以下の論理式が恒真であれば、変更前と変更後の仕様ルールは等価である。

$$\exists \boldsymbol{y}, j, \forall \boldsymbol{x}, i \cdot i \in ERS \wedge Cond_i(\boldsymbol{x}) \wedge \mathbb{C} \rightarrow j \in RS' \wedge Res_i(\boldsymbol{y}) \wedge Cond'_j(\boldsymbol{x}) \wedge Res'_j(\boldsymbol{y}) \wedge \mathbb{C}' \quad (4)$$

ここで、プライム(‘)付の記号は変更後のルールに関するものであり、プライム(‘)のないものは変更前のルールに関するものである。

3.2 機能要件

等価性検証の機能要件を以下の通り、設定した。

【要件１】等価であるか否かを判定できること

【要件２】等価でない場合、等価でないルールを全て特定できること

要件１は等価性検証の最低限の要件である。要件２は、等価でない場合に、開発者がどこに原因があるかを特定する際に必要な機能である。

3.3 処理方式

3.3.1 処理概要

等価性の検証は、変更前の仕様ルールセット及び条件制約ルールセット、等価性維持仕様ルールセット、変更後の仕様ルールセット及び条件制約ルールセットをインプットとして、式(4)が常に成り立つことを示せばよい。充足可能性問題に帰着させると、式(4)の否定が UNSAT であればよいが、これでは、要件１を満たせても要件２を満たすことができない。また、式(4)は、量化記号を含むため、単純に SMT ソルバに式(4)を与えても、ほとんどの場合、充足可能性を判定できない。

そこで、個々の等価性維持仕様ルールについて、段階を追って処理を進めることで、量化記号の問題と要件２への対応を行う。具体的には、等価性維持仕様ルール１つ１つについて再帰的に処理を行う。１つの等価性維持仕様ルールiに着目すると、式(4)は、下式のようになる。

$$\exists \boldsymbol{y}, j, \forall \boldsymbol{x} \cdot Cond_i(\boldsymbol{x}) \wedge \mathbb{C} \rightarrow j \in RS' \wedge Res_i(\boldsymbol{y}) \wedge Cond'_j(\boldsymbol{x}) \wedge Res'_j(\boldsymbol{y}) \wedge \mathbb{C}' \quad (5)$$

式(5)において、$Cond_i(\boldsymbol{x}) \wedge \mathbb{C}$を真にする全ての入力$\boldsymbol{x}$について、式(5)の後件(consequent)を真にする出力$\boldsymbol{y}$と変更後仕様ルールjが存在することを示せばよい。式(5)ではまだ量化記号が存在するため、SMT ソルバに与える論理式に量化記号が含まれないように、以下の３つのステップで等価性検証の処理を行う。

(1) 対応関係の解析

式(5)の後件を真にする、すべての変更後仕様ルールjを特定する。これをすべての等価性維持仕様ルールについて実施し、等価性維持仕様ルールと変更後仕様ルールの対応関係を明確にする。

(2) 包含関係の解析

$Cond_i(\boldsymbol{x}) \wedge \mathbb{C}$を真にする全ての入力$\boldsymbol{x}$について、(1)の処理で特定した変更後仕様ルールのいずれかが真になることを示す。これは、式(5)の前件(antecedent)を満たす入力$\boldsymbol{x}$の集合が、式(5)の後件を満たす入力$\boldsymbol{x}$の集合に含まれることを示せばよい。

(3) 等価性の判定

(1)(2)の処理結果に基づき、等価であるか否かの判定、および、等価でない場合、等価でない原因を出力する。

図 1 は、以上を踏まえた等価性検証の処理フローを示している。なお、N は、変更前仕様ルールの数を表す。図 1 に示す処理では、等価性維持仕様ルールに限定せず、すべての変更前仕様ルールについて処理を行うようにしている。これは、等価性維持仕様ルール以外の変更前仕様ルールについても、開発者が、変更後仕様ルールとの対応関係や包含関係を把握できるようにするためである。

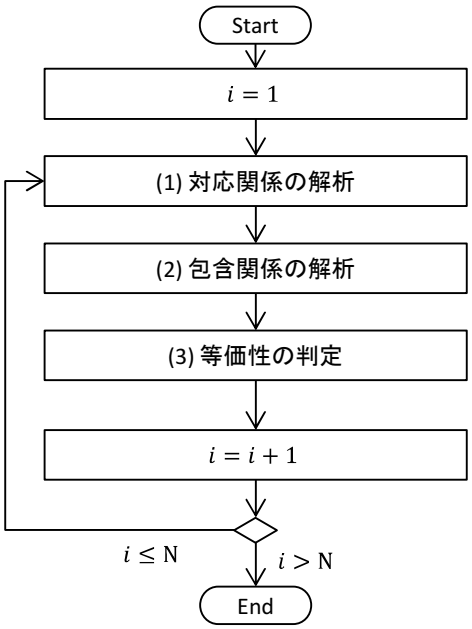

図 1　等価性検証の処理フロー

以下では、処理(1)から(3)の詳細について、述べる。

3.3.2 対応関係の解析

対応関係の解析では、式(5)の後件を真にする、すべての変更後仕様ルールjを特定する。これをすべての等価性維持仕様ルールについて実施し、等価性維持仕様ルールと変更後仕様ルールの対応関係を明確にする。

式(5)の後件が真になる場合、変更後仕様ルールjは、等価性維持仕様ルールの条件部との論理積が充足可能であり、結果部との論理積が充足可能となる。このような関係にある変更後仕様ルールjを見つければよい。図 2 は、これを実現するための対応関係の解析の処理フローを示している。

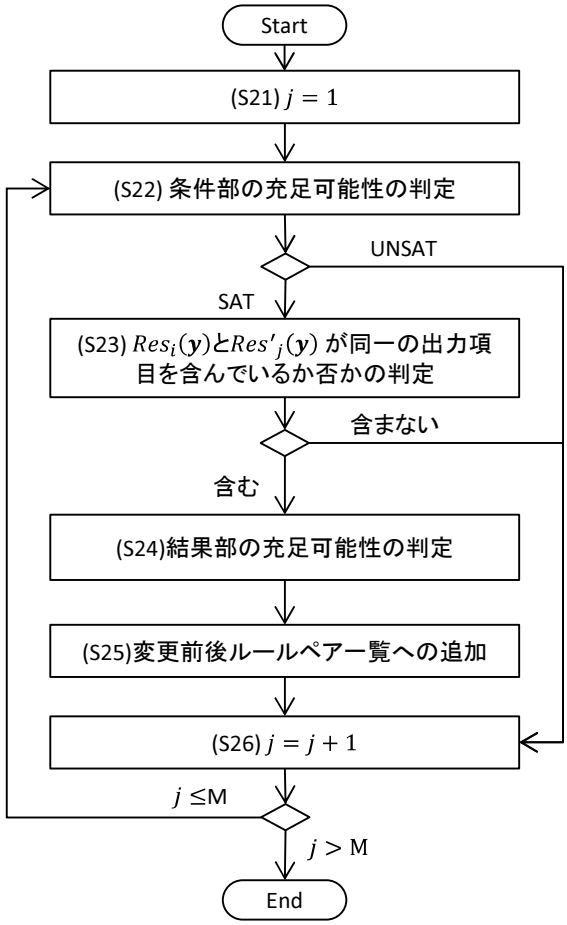

図 2　対応関係の解析の処理フロー

図 2 を参照して、対応関係の解析の処理を説明する。まず、パラメータ j の値を「1」で初期化する(S21)。jは、変更後仕様ルールの識別番号を表す。次に、変更前と変更後の仕様ルールの条件部の充足可能性を下式により判定する(S22)。充足可能性の判定には、SMT ソルバを利用する。充足可能な場合、変更前仕様ルールiの条件部と変更後仕様ルールjの条件部を同時に満たす入力の組合せが存在することを意味する。

$$Cond_i(\boldsymbol{x}) \wedge Cond'_j(\boldsymbol{x}) \wedge \mathbb{C} \wedge \mathbb{C}' \quad (6)$$

続いて、$Res_i(\boldsymbol{y})$と$Res'_j(\boldsymbol{y})$が同一の出力項目を含んでいるか否かを判定する(S23)。同一の出力項目を含んでいない場合、互いに影響を及ぼさないため、変更前の仕様ルールiと変更後の仕様ルールjの間には対応関係はないものとみなす。同一の出力項目を 1 つでも含んでいる場合は、変更前と変更後の仕様ルールの結果部の充足可能性を下式により判定する(S24)。充足可能な場合、変更前の仕様ルールiと変更後の仕様ルール j は、整合しており、充足不能な場合は、相反する結果を返すこととなるため、不整合となる。

$$Res_i(\boldsymbol{y}) \wedge Res'_j(\boldsymbol{y}) \quad (7)$$

次に、そのルールペアに関する情報（以下、「変更前後ルールペア」という。）を変更前後ルールペア一覧に追加する(S25)。表 1 は、変更前後ルールペア一覧の例である。変更前後ルールペアは、変更前仕様ルールＩＤ、変更後仕様ルールＩＤ、および、変更前と変更後の仕様ルールの充足可能性の判定結果を保持している。

表 1　変更前後ルールペア一覧の例

変更前仕様ルールＩＤ	変更後仕様ルールＩＤ	結果部の充足可能性
1	1	SAT
1	2	SAT
2	2	UNSAT
・・・	・・・	・・・

S22 から S25 までの処理をすべての変更後仕様ルールについて実施することで対応関係の明確にする。S26 は、全ての変更後仕様ルールを処理するためのパラメータ j のカウンタアップ処理である。また、M は、変更後仕様ルールの数である。

以上の処理により、変更前と変更後の仕様ルールの対応関係が明確となり、表 1 を参照することで、どの変更前の仕様ルールと変更後の仕様ルールが対応関係にあるかに加え、整合しているか、もしくは、不整合となっているかの情報を得ることができる。

3.3.3 包含関係の解析

包含関係の解析では、$Cond_i(\boldsymbol{x}) \wedge \mathbb{C}$を真にする全ての入力$\boldsymbol{x}$について、対応関係の解析処理で特定した変更後仕様ルールのいずれかが真になることを示す。これは、式(5)の前件(antecedent)を満たす入力$\boldsymbol{x}$の集合が、式(5)の後件を満たす入力$\boldsymbol{x}$の集合に含まれることを示せばよい。

図 3 は、これを実現するための包含関係の解析の処理フローを示している。

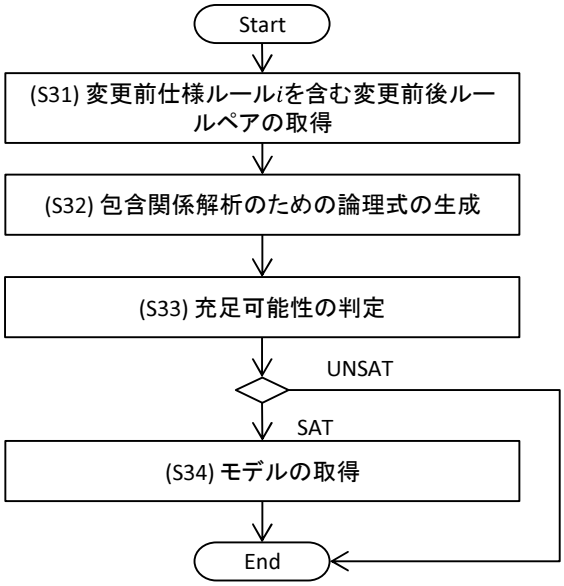

図 3　包含関係の解析の処理フロー

図 3 を参照して、包含関係の解析の処理を説明する。まず、変更前後ルールペア一覧を参照し、変更前仕様ルール ID が i を含む変更前後ルールペアを全て取得する(S31)。次に、変更前仕様ルールと変更後仕様ルールの包含関係を解析するための論理式を生成する(S32)。包含関係をチェックすることで、変更前仕様ルールを満たす入力と出力のすべての組合せが、変更後仕様ルールにおいても満たすことを確認できる。具体的には、S32 では、下式を生成する。

$$\neg\left(Cond_i(\boldsymbol{x}) \wedge \mathbb{C} \rightarrow \bigwedge_{p\in \mathrm{RP}_i}\left(\bigvee_{j\in \mathrm{RSP'}_{ip}} Cond'_j(\boldsymbol{x})\right) \wedge \mathbb{C}'\right) \qquad (8)$$

ここで、RP_iは、変更前仕様ルールi の結果部に含まれる出力項目の集合であり、$\mathrm{RSP'}_{ip}$は、変更前仕様ルールi と対応関係にある変更後仕様ルールのうち、出力項目p を含む変更後仕様ルールの ID の集合である。なお、全ての変更前仕様ルールおよび変更後仕様ルールの出力項目が同じ場合、下式でも良い。

$$\neg\left(Cond_i(\boldsymbol{x}) \wedge \mathbb{C} \rightarrow \bigvee_{j\in \mathrm{PRS}_i} Cond'_j(\boldsymbol{x}) \wedge \mathbb{C}'\right) \qquad (9)$$

ここで、PRS_iは、変更前仕様ルールi と対応関係にある変更後仕様ルールの ID の集合である。

続いて、S32 で生成した論理式の充足可能性を判定する(S33)。充足不能な場合、変更前仕様ルールを満たす入力と出力のすべての組合せが、変更後仕様ルールにおいても満たすことを意味する。充足可能な場合、変更後仕様ルールに漏れがあることを意味し、S34 にて、漏れている入力$\boldsymbol{x}$の組合せを取得する。

以上の処理により、変更前仕様ルールと変更後仕様ルールの包含関係を明確にする。

3.3.4 等価性の判定

等価性の判定では、対応関係の解析結果、および、包含関係の解析結果に基づき、等価であるか否かの判定、および、等価でない場合、等価でない原因を出力する。

変更前仕様ルールが等価であるのは、変更前仕様ルールが等価性維持仕様ルールであり、かつ、対応関係にある変更後仕様ルールに式(7)が UNSAT であるものを含まず、かつ、式(8)または(9)が UNSAT の場合のみである。変更前仕様ルールが等価性維持仕様ルールでない場合は、等価性判定の対象外となる。

また､変更前仕様ルールが等価性維持仕様ルールであり、かつ、対応関係にある変更後仕様ルールに式(7)が UNSAT であるものを含む場合、そのような関係にある変更前仕様ルールと変更後仕様ルールのペアを出力する。また、変更前仕様ルールが等価性維持仕様ルールであるか否かに係らず、式(8)または(9)が SAT の場合には、SAT となる入力$\boldsymbol{x}$の組合せを出力する。

3.4 検証モデリング方式

検証を行う上では、検証エンジンのインプットとなる情報を作成する検証モデリングが必要である。前項で述べた通り、等価性の検証では、充足可能性の判定処理が含まれる。充足可能性の判定には、SMT ソルバを利用できる。そのため、等価性の検証では、検証エンジンとして SMT ソルバを利用する。SMT ソルバを実行する上では、SMT ソルバが解釈可能な入力形式に充足可能性問題を変換する必要がある。多くの SMT ソルバは、SMT-LIB2 6)という入力形式に対応しており、本研究では、汎用性を考慮し、検証モデリングでは、SMT-LIB2 を利用することとした。

例を用いて、SMT-LIB2 について説明する。

表 2、表 3 は、それぞれ変更前仕様ルールセットと変更後仕様ルールセットを表す。変更前仕様ルールセットには５つの変更前仕様ルールが含まれ、ID が、V1_1、V1_2、V1_4 である変更前仕様ルールが等価性維持仕様ルールである。なお、この例では、条件制約ルールは含まれない。

表 2　変更前仕様ルールセット

ID	条件部	結果部	等価性
V1_1	Age ≤ 35 ∧ Rank ≠ Gold	Discount = 5	YES
V1_2	$35 < Age < 60$ ∧ Rank = Bronze	Discount = 10	YES
V1_3	$35 < Age < 60$ ∧ Rank = Silver	Discount = 15	NO
V1_4	Age ≥ 60	Discount = 20	YES
V1_5	Rank = Gold	Discount = 20	NO

表 3　変更後仕様ルールセット

ID	条件部	結果部
V2_1	Age ≤ 35 ∧ Rank ≠ Gold	Discount = 5
V2_2	$35 < Age < 60$ ∧ Rank ≠ Gold	Discount = 10
V2_3	Age ≥ 60	Discount = 15
V2_4	Age≤60∧Rank = Gold	Discount = 10

図 4 は、表 2、表 3 の例について、全ての変更前仕様ルールと変更後仕様ルールのペアについて、式(6)の充足可能性を判定するための SMT-LIB2 の例を示している。図 4 では、最初に利用するロジックを、「set-logic」により指定し、続いて、入力項目、出力項目、変更前仕様ルール、変更後仕様ルール、式(6)の充足可能性判定式を記述している。SMT-LIB2 では、push コマンドと pop コマンドを利用することで、複数の充足可能性問題を記述できる。

図 4　条件部の充足可能性判定のための SMT-LIB2 の例

```
(set-logic QF_LIA)

; Input
(declare-fun Age () Int)
(assert (>= Age 0))

(declare-fun Rank () Int)
(assert (and (>= Rank 1)(<= Rank 3)))
(define-fun Bronze () Int 1)
(define-fun Silver () Int 2)
(define-fun Gold () Int 3)

; Output
(declare-fun Discount () Int)

; Pre-change rules
(define-fun CondV1_1 () Bool (and (<= Age 35) (not (= Rank Gold))))
(define-fun CondV1_2 () Bool (and (< 35 Age) (< Age 60) (= Rank Bronze)))
(define-fun CondV1_3 () Bool (and (< 35 Age) (< Age 60) (= Rank Silver)))
(define-fun CondV1_4 () Bool (>= Age 60))
(define-fun CondV1_5 () Bool (= Rank Gold))

; Post-change rules
(define-fun CondV2_1 () Bool (and (<= Age 35) (not (= Rank Gold))))
(define-fun CondV2_2 () Bool (and (< 35 Age) (< Age 60) (not (= Rank
Gold))))
(define-fun CondV2_3 () Bool (>= Age 60))
(define-fun CondV2_4 () Bool (and (<= Age 60) (= Rank Gold)))

; Evaluate satisfiability of condition part
(push 1)
(assert (and CondV1_1 CondV2_1))
(check-sat)
(pop 1)

(push 1)
(assert (and CondV1_1 CondV2_2))
(check-sat)
(pop 1)

(push 1)
(assert (and CondV1_1 CondV2_3))
(check-sat)
(pop 1)

(push 1)
(assert (and CondV1_1 CondV2_4))
(check-sat)
(pop 1)

・・・
```

4. 事例適用結果

4.1 題材の概要

題材として利用した行政事務上の業務ルールは、条件表という表形式で記述されている。図 5 に条件表の例を示す。

		1	2	3	4	5	6	7	8	9	10	11	12
判定	Condition 1	Y	N	N	N	N	N	N	N	N	N	N	N
	Condition 2		Y	Y	N	N	N	N	N	N	N	N	N
	Condition 3				Y	Y	Y	Y	Y	Y	Y	Y	N
	Condition 4				Y	Y	Y	Y	N	N	N	N	Y
	Condition 5		Y	N	Y	Y	N	N	Y	Y	N	N	Y
	Condition 6				Y	N	Y	N	Y	N	Y	N	Y
処理	Param 1												
	Value 11	○	○		○				○				○
	Value 12			○		○	○	○		○	○	○	
	Param 2												
	Value 21			○									
	Value 22	○	○		○				○				○

図 5　条件表の例

条件表は、判定条件を記述するエリア（判定欄）と処理内容を記述するエリア（処理欄）から構成され、各列が 1 つの仕様ルールに対応する。判定条件の「Y」は、当該判定条件が真であることを表し、「N」は偽であることを表す。空欄はドントケアを表す。また、処理については、「○」が記載された値がパラメータに代入されることを表す。いずれの値にも「○」が記載されていない場合には、いずれの値も代入しないことを表す。例えば、３列目の仕様ルールは、以下を意味する。

If ¬Condition1∧Condition2∧¬Condition5
then Param1 = Value11 ∧ Param2 = Value21

事例適用の対象とした条件表の変更前と変更後の情報を表 4 に示す。

表 4　条件表の変更前と変更後の情報

#	項目	変更前仕様ルール	変更後仕様ルール
1	ルール数	114	18
2	入力項目数	12	7
3	判定条件数	17	6
4	出力項目数	2	2

4.2 提案方式の適用方法

提案方式の適用の流れは、以下の通りである。

(1) **条件表からの入出力項目の抽出（手作業）**

条件表の判定条件および処理から入出力項目を抽出し、SMT-LIB2 形式で記述

(2) **判定条件・処理の抽出（手作業）**

(1)で抽出した入力項目を活用し、判定条件および処理を SMT-LIB2 形式で記述

(3) **仕様ルールの抽出（自動）**

条件表の「Y」「N」「○」の情報をもとに、条件表から SMT-LIB2 形式で出力する Excel マクロを作成し、仕様ルールを自動抽出

(4) **等価性判定の論理式の生成（自動）**

等価性判定の SMT-LIB2 形式の論理式を生成する Excel マクロを作成し、等価性判定の論理式を自動生成

(5) **SMT ソルバによる充足可能性判定（自動）**

(4)までで作成した SMT-LIB2 で記述したファイルを SMT ソルバにより充足可能性を判定。

(6) **充足可能性判定結果の解析（手動）**

(5)の充足可能性判定結果から、等価性を確認。

なお、提供された題材については、変更前仕様ルールのどれが等価であるべきかどうかの情報がなかったため、等価なルールとそうでないルールの判定を行った。

4.3 適用結果

提案方式を適用した結果、変更前仕様ルール 114 件中、17 件が変更後仕様ルールと非等価であることを抽出した。開発者に確認したところ、仕様変更した箇所であり、問題ないことを確認した。

また、等価性検証に係る作業時間としては、約 11 時間 30 分であり、内 6 時間は仕様理解に要した時間である。なお、評価対象の題材については、等価性検証の他、整合性、完全性、実行可能性についても検証を行い、問題ないことを確認した。

開発担当者からは、「現状、見直しは一人の有識者に依存せざるを得ない状況であったが、今回の検証で不備がないことを担保でき助かった」とコメント頂いた。

5. おわりに

本稿では、変更前と変更後の２つのバージョンのルールに着目し、ルールの変更前後で、入力と出力の関係が維持されるべき箇所が維持されているかの確認を支援する等価性検証方式を提案した。本方式では、SMT ソルバを活用して、変更後において入力と出力の対応関係が維持されるべき変更前のルール（等価性維持仕様ルール）と変更後仕様ルールの対応関係、および、条件部の包含関係を解析し、その結果に基づき、等価性を判定する。従来のサンプリングテストに基づく方式と比較して、網羅的にチェックできる点が優位である。

また、本方式を行政事務上の業務ルールの見直し案件を事例として適用した結果、変更前仕様ルール 114 件中 17 件が変更後仕様ルールと非等価であることを抽出した。これは期待通りの結果であり、提案方式により等価性を判定できることを確認した。

今回の事例適用では、手作業で検証モデリングを実施しており、検証に多くの時間を要した。今後は、提案方式のツール化を検討し、作業の効率化を進める。

参考文献

1) 井上英明：「超高速開発」が日本を救う―サムスンはすでに始めている、日経コンピュータ、2012 年 3 月 15 日号
2) Bruno Berstel-Da Silva: Verification of Business Rules Programs, Springer, 2014
3) Bruno Berstel, Michel Leconte, Using Constraints to Verify Properties of Rule Programs, ICST. IEEE Computer Society, 2010, pp.349-354.
4) 伊藤信治他：SAT ソルバを活用した決定表作成・検証方式、電子情報通信学会ソフトウェアサイエンス研究会、2014/3
5) Barbara von Halle: Business Rules Applied, John Wiley & Sons, 2001/9
6) Clark Barrett, Aaron Stump and Cesare Tinelli, “The SMT-LIB Standard: Version 2.0”, 2012/9, http://smtlib.cs.uiowa.edu/papers/smt-lib-reference-v2.0-r12.09.09.pdf