情報処理学会論文誌 Vol. 50 No. 2 701–1241 (Feb. 2009)

複合 Web サービスの実行可能性予測

田仲正弘[†1] 石田亨[†1]

標準化されたインターフェースを持つ原子 Web サービスに基づいて複合 Web サービスを構成することで，呼び出し先アドレスを設定するだけで，使用する原子 Web サービスを選択し，アプリケーションに応じた機能を実現できる．しかし，原子 Web サービスの選択によっては，原子 Web サービスの実行に失敗して複合 Web サービスの実行が停止し，すでに実行された原子 Web サービスの実行コストが無駄になる．従来の研究では，モデルチェッキングなどを用い，複合 Web サービスがあらゆる要求メッセージに対して正常に実行可能であることを検証する手法が提案されてきた．しかし，従来の研究には，次のような問題がある．1) 検証によってエラーが起こりうるとわかっても，原子/複合 Web サービスの修正が困難である．2) 原子 Web サービスの完全な仕様が公開されておらず，検証が行えないことがある．そこで本研究では，要求メッセージごとに複合 Web サービスの実行可能性を予測する．実行可能と予測された要求メッセージでのみ実行することで，無駄な原子 Web サービスの実行を抑止する．さらに，予測に用いる仕様が不明な場合に，実行の成否に基づいて仕様を学習し，予測を改善する．これを実現するため，形式的仕様記述による Web サービスのモデル化を行い，定理証明器を用いた要求メッセージごとの実行可能性予測を実現した．また，制約獲得アルゴリズムを適用し，原子 Web サービスの入力仕様を学習し，実行可能性の予測を改善した．

Predicting Executability of Composite Web Services

Masahiro Tanaka[†1] and Toru Ishida[†1]

Constructing a composite Web service using atomic Web services which have standardized interfaces can increase the reusability of the composite Web service. The developer of an application can easily select atomic Web services which have appropriate functionalities only by setting endpoints. However, the selection might allow a request message to cause failures of execution of atomic Web services and the cost of the execution goes to waste. Previous works proposed methods of validation, such as model checking, that check if a composite service is executable for any request message. But the previous works have the following problems: 1) To modify atomic/composite services is usually impossible even if execution of the services might fail. 2) Complete specifications of atomic Web services which are required for the validation are often unavailable. To solve the problems, we propose a method for predicting executability of composite Web services for each request message. This prevents the waste of cost of executing atomic Web services. We modeled atomic/composite services in formal specification and used a theorem prover to check if a given request message is valid. Moreover we improved the prediction by applying constraint acquisition algorithm to learn input specification of atomic Web services.

1. はじめに

近年様々な組織により，Web サービスのインターフェースの標準化が進められている．そのため，以下の開発の手順により，複合 Web サービスの再利用性を向上させることが重要になる．複合 Web サービスの設計者は，WS-BPEL[1] や OWL-S[2] を用い，インターフェースのみを定義した原子 Web サービス (抽象原子 Web サービス) に基づいて，複合 Web サービスを構成する (抽象複合 Web サービス)．原子 Web サービスのインターフェースが標準化されていれば，アプリケーション開発者は，各原子 Web サービスの呼び出し先アドレスを設定するだけで，実際に利用する原子 Web サービス (具象原子 Web サービス) を選択し，アプリケーションに必要な機能を持った複合 Web サービス (具象複合 Web サービス) を開発できる．

例えば言語グリッド[3] の場合，様々な組織によって開発された機械翻訳や辞書などの各種の言語サービスを Web サービスとして提供している．これらの Web サービスは，言語サービスオントロジー[4] に基づいて，種類ごとにインターフェースが標準化されている．そのため，これらの言語サービスを連携した複合 Web サービスを利用するアプリケーションの開発者は，各抽象原子 Web サービスの呼び出し先アドレスを設定するだけで，サポートする言語などの異なる様々な具象複合 Web サービスを容易に利用できる．

しかし，Web サービスのインターフェースを定義する標準的な記述である WSDL では，各具象 Web サービスの仕様の差異を表現できない．例えば，ホテル予約を行う Web サービスが，要求メッセージとして予約希望日を受け取るとする．何日前から予約を受け付けるかは，具象原子 Web サービスごとに異なるが，このような仕様は WSDL に記述されない．したがって，ホテル予約 Web サービスの WSDL による記述に基づいて抽象複合 Web サー

†1 京都大学情報学研究科社会情報学専攻
Department of Social Informatics, Kyoto University

ビスを構成し，呼び出し先アドレスの設定によって具象 Web サービスを選択すると，与えられた要求メッセージによってはエラーを起こして実行が失敗することがある．入力仕様に適合しない要求メッセージを与えられたことで一部の原子 Web サービスが実行に失敗すると，それ以前に実行された原子 Web サービスの実行コストが無駄になるという問題がある．

本研究では，そのような複合 Web サービスの実行失敗時における，無駄な原子 Web サービスの実行を抑止することを目的とする．従来の研究では，モデルチェッキングなどを用い，複合 Web サービスが，あらゆる要求メッセージに対して正常に実行可能であることを検証する手法が提案されてきた[5)–7)]．しかし，先に述べた開発手順のように，具象原子 Web サービス提供者・抽象複合 Web サービス設計者・アプリケーション開発者がそれぞれ別である場合，従来の手法による複合 Web サービスの検証には，以下に述べる問題がある．

- アプリケーション開発者による検証で，具象複合 Web サービスがエラーを起こしうると判断されても，具象原子 Web サービス・抽象複合 Web サービスの修正が困難である．
- 使用する具象原子 Web サービスの完全な仕様が公開されているとは限らず，アプリケーション開発者が，具象複合 Web サービスの検証を行えないことがある．

本研究では，これらの問題を解決するため，以下のアプローチを取る．1 点目の問題の解決のため，複合 Web サービスに与えられる要求メッセージそれぞれについて，その複合 Web サービスの実行可能性を予測する．実行不可能と予測された場合には，実行を中止する．これにより，あらゆる要求メッセージに対して正常に実行できる具象複合 Web サービスが構築できない場合に，実行可能と予測された要求メッセージについてのみ実行を行い，無駄な原子 Web サービスの実行を抑止する．さらに，2 点目の問題の解決のため，予測に用いる具象原子 Web サービスの仕様が不明な場合には，まず実行を試み，その成否に基づいて仕様を学習することで，以降の予測を改善する．

これを実現するため，本研究では，形式的仕様記述による原子/複合 Web サービスのモデルを定義し，複合 Web サービス中の全ての原子 Web サービスが実行可能かどうかを，定理証明器によって判断する手法を提案する．また，制約獲得アルゴリズム[8)] を用い，各具象原子 Web サービスの入力仕様を制約として学習することで，形式的仕様記述による Web サービスのモデルを更新し，実行可能性の予測を改善する．

以降では，2 章で形式的仕様記述による Web サービスのモデルについて，3 章で制約獲得アルゴリズムによる入力仕様の学習について，それぞれ説明する．さらに，4 章で機械翻訳・専門用語辞書・形態素解析などの原子 Web サービスからなる複合 Web サービスに適用した結果を示す．5 章で関連研究を紹介した後，6 章で本研究での提案手法についてまとめを述べる．

2. 形式的仕様記述による Web サービスのモデル化

本章では，実行可能性予測のための，形式的仕様記述による Web サービスのモデル化について説明する．

本研究では，形式的仕様記述に，代数的仕様記述言語[9)] の一つである CafeOBJ[10)] を使用する．Web サービスの場合，その完全な仕様が公開されていないことも多いが，CafeOBJ のような代数的仕様記述言語では，仕様を任意の既知の範囲で記述できる．

2.1 原子 Web サービスの仕様記述

複合 Web サービスの実行可能性を判断するためには，各原子 Web サービスについて，以下の情報が必要となる．

- ある要求メッセージで実行可能か (入力仕様)
- ある要求メッセージに対して，どのような応答メッセージが得られるか (入出力関係)

前者により，複合 Web サービス中の各原子 Web サービスに与えられる要求メッセージで，それぞれの原子 Web サービスが実行可能であるかどうかを判断できる．全ての原子 Web サービスが実行可能であるとき，複合 Web サービスは実行可能である．後者は，ある原子 Web サービスの結果が別の原子 Web サービスに与えられるような複合 Web サービスにおいて，複合 Web サービスに与えられた要求メッセージから，各原子 Web サービスに与えられる要求メッセージを求めるのに必要である．

本研究では，これらを形式的仕様記述で表現する Web サービスのモデルを提案することで，定理証明器を用いた実行可能性の予測を実現する．

本研究では，原子 Web サービスを代数的仕様記述におけるモジュールとしてモデル化する．入力仕様と入出力関係は，それぞれ下記の 2 つのオペレーションによって表現される．

`domain-`*service-name*　入力仕様を表現する．要求メッセージの値を受け取り，実行可能性を表す真偽値を返す．true が返されれば実行可能であり，false が返されれば実行不可能であるとする．

`execute-`*service-name*　入出力関係を表現する．要求メッセージの値を受け取り，応答メッセージの値を返す．

ただし，Web で公開される Web サービスを使用する場合，入力仕様・入出力関係のいずれについても，完全に既知であるとは限らない．また一般に，ある要求メッセージに対してどのような応答メッセージが得られるかは，Web サービスが実行される以前は不明であり，

完全な入出力関係を仕様として記述することはできない．そこで，要求メッセージと応答メッセージの要素の値や型について，オペレーションに関する公理として，既知の範囲で制約を記述する．これらのオペレーションの定義は，Problem Theory[11] におけるプログラムの仕様の表現を，特定の入出力値に対する制限を記述するように変更したものである．

以下で，実際の仕様記述について，**図 1** に示したある機械翻訳 Web サービスの仕様を例に説明する．

```
 1: mod TRANSLATOR {
 2:   pr(LANGUAGE + TRANSLATOR-REQUEST + TRANSLATOR-RESPONSE)
 3:
 4:   op domain-translator : TranslatorRequest -> Bool
 5:   op execute-translator : TranslatorRequest
                                -> TranslatorResponse
 6:
 7:   var e : TranslatorRequest
 8:
 9:   -- Source language must be English or Japanese
10:   eq domain-translator(e) =
11:       1*(e) == english or 1*(e) == japanese .
12:   -- Language of result is specified by target language
13:   eq get-language(execute-translator(e)) = 2*(e).
14: }
```

図 1　機械翻訳 Web サービスの仕様記述
Fig. 1　Specification of a machine translation service.

初めに，使用されるデータ型や要求/応答メッセージの定義がインポートされる (2 行目)．この機械翻訳 Web サービスへの要求メッセージは翻訳元言語・翻訳先言語・翻訳される文字列からなる 3-tuple であり，応答メッセージは翻訳された文字列であるとする．次に，入力仕様と入出力関係を表現するオペレーションが宣言される (4-5 行目)．これらのオペレーションの内容は，後に続く公理で定義される (10-11, 13 行目)．1 つ目の公理は，要求メッセージの 3-tuple の 1 つ目の値 (翻訳元言語) が，日本語または英語でなくてはならないことを示している．2 つめの公理は，3-tuple の 3 つめの値 (翻訳される文字列) は，3-tuple の 2 つめの値 (翻訳先言語) の言語に翻訳されることを示している．“`n*`” は，N-tuple から `n` 番目の値を取り出す，N-tuple 上に定義されたオペレーションである．

2.2　複合 Web サービスの仕様記述

複合 Web サービス全体の実行可能性を予測するには，原子 Web サービスと同様に複合 Web サービスの仕様記述が必要になる．複合 Web サービスの仕様記述は，原子 Web サービスの仕様記述を組み合わせて構築される必要がある．これにより，使用する原子 Web サービスが変更された場合にも，その原子 Web サービスの仕様記述を置き換えるだけで，複合 Web サービスの仕様記述も更新される．

これを実現するため，本研究では，複合 Web サービスの制御構造の性質に基づく制約を形式的仕様記述における公理として表現し，制御構造のブロックのネスト構造に基づいて，ボトムアップに原子 Web サービスの仕様記述を組み合わせる手法を示す．

OWL-S や WS-BPEL によって記述される複合 Web サービスでは，制御構造のブロックは，その内部に原子 Web サービスあるいは別の制御構造のブロックを含むネスト構造を持つ．そこで，制御構造のブロックに相当する仕様記述を，そのブロックに含まれる原子 Web サービス仕様記述あるいは別の制御構造のブロックに相当する仕様記述に基づいて再帰的に定義する．

これには，原子 Web サービスと制御構造のブロックで，統一的な仕様記述が必要となる．そこで，制御構造のブロックを 1 つの Web サービスと見なし，原子 Web サービスと同様に要求/応答メッセージ・入力仕様・入出力関係を考える．他の全ての制御構造のブロックまたは原子 Web サービスを包含するブロックが，複合 Web サービス全体に相当する．

そのため，制御構造のブロックに相当する仕様記述を，原子 Web サービスの仕様記述と同様の形式で定義する．すなわち，入力仕様と入出力関係を表現する 2 種類のオペレーションを定義する．

また，原子 Web サービスの場合に加えて制御構造のブロックの仕様記述の場合に考慮するべき要素として，原子 Web サービスまたは制御構造のブロック間のデータフローと，制御構造の性質に基づく制約がある．データフローは，要求/応答メッセージの各要素の対応を定義する．制御構造の性質に基づく制約としては，例えば Sequence 制御構造の場合，ブロック中の原子 Web サービスは順に実行され，またいずれの原子 Web サービスも実行可能でなくてはならない．また，While 制御構造では，条件が満たされる限り，ブロック中の原子 Web サービスは反復実行されるため，その原子 Web サービスは実行可能でなくてはならないが，条件が満たされない状態では，ブロック中の原子 Web サービスは実行可能でなくても良い．

データフローや制御構造の性質に基づく制約は，制御構造のブロックの仕様記述において，入力仕様と入出力関係を表現するオペレーションに関する公理として定義できる．

以下で，**図 2** を例に説明する．図 2 は，点線で示した Sequence 制御構造のブロックに，2 つの機械翻訳 Web サービス Machine Translator A, B を含む複合 Web サービスを示している．この複合 Web サービスは，Machine Translator A によって文字列を別の言語 (中間言語) に翻訳したあと，Machine Translator B によって元の言語に再び翻訳することで，

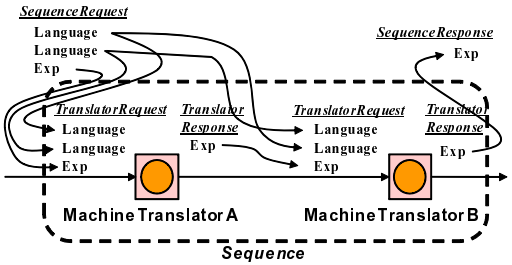


図 2　複合 Web サービス中のブロックとデータフロー
Fig. 2　Dataflow in a control block.

中間言語への翻訳の品質を見積もることを目的としたものである．

ブロック及び原子 Web サービスの右上と左上には，それぞれ要求メッセージと応答メッセージ及びそれらの要素を示している．この複合 Web サービスへの要求メッセージは，翻訳元言語・中間言語・翻訳される文字列からなる 3-tuple となっている．また応答メッセージは，翻訳結果の文字列からなる 1-tuple となる．N-tuple の要素をつなぐ線は，データフローを示している．

図 3 に，図 2 のブロックの仕様記述を示す．オペレーションの宣言は原子 Web サービスと同様であるが，制御構造に基づく制約が公理に記述される．1 つめの公理は，ブロックの入力仕様に相当するもので，2 つの機械翻訳 Web サービスの入力仕様に相当するオペレーションの連言を取ることで，与えられた要求メッセージによって両方の Web サービスが実行可能でなければならないことを示している．また 2 つめの公理は，ブロックの応答メッセージは，Machine Translator B の応答メッセージとなることを示している．さらに，N-tuple の操作による，各オペレーションへの引数の生成により，データフローが表現される．

OWL-S や WS-BPEL では，条件分岐・並列実行等の制御構造が提供され，それらの多くは，ここで述べた方法によって定義可能であることを確認した．ただし，OWL-S で提供される，複数の Web サービスの任意の順序での実行を記述する制御構造 Any Order は，ここで述べた方法で表現することは困難であった．

複合 Web サービスの実行可能性予測を行うには，引数に複合 Web サービスへの要求メッセージを与えて，定理証明器によって複合 Web サービス全体に相当するブロックの `domain-`*service-name* を簡約する．簡約の結果，`true` が得られれば実行できる可能性があると予測され，実行が開始される．`false` が得られれば，実行は中止される．

```
mod SEQUENCE {
...
  op domain-sequence : SequenceRequest -> Bool
  op execute-sequence :
            SequenceRequest -> SequenceResponse
...
  var request : SequenceRequest
...
  eq domain-sequence(request) =
   domain-translator-a(
    1*(request), 2*(request), 3*(request))
   and domain-translator-b(
    2*(request), 1*(request),
     execute-translator-a(
      1*(request), 2*(request), 3*(request))) .

  eq execute-sequence(request) =
   execute-translator-b(
    2*(request), 1*(request),
    execute-translator-a(
      1*(request), 2*(request), 3*(request))) .
}
```

図 3　図 2 のブロックの仕様記述例
Fig. 3　Specification of the block in Fig. 2.

ただし，複合 Web サービス実行の過程で，すでに実行された原子 Web サービスの応答メッセージを用いると，より正確な予測が可能になる．そのため，原子 Web サービスの実行が完了する度に，得られた応答メッセージに基づいて，未実行のブロックや原子 Web サービスの `domain-`*service-name* を簡約する．いずれかの簡約の結果 `false` が得られた場合には，以降の実行が中止される．

複合 Web サービス実行の過程での予測により，予測を行わない場合に比べて，実行中に必要な処理が増えることになる．しかし，定理証明器での簡約に要する時間は，Web 上の通信による XML データの交換が必要となる Web サービス実行に要する時間と比較して小さく，パフォーマンス上の問題とはなりにくい．また，原子 Web サービスの実行ごとに利用料などのコストが発生する場合，無駄な実行の抑止は特に有用である．

3. 入力仕様の学習

Web で公開される Web サービスの場合，その Web サービスの提供者が完全な仕様を公開しているとは限らない．そのため，2 章で述べた形式的仕様記述による実行可能性予測が，十分に機能しない可能性がある．そこで，実行の成否に基づいて，具象原子 Web サービスごとに，入力仕様を学習する．

前述の Web サービスのモデルでは，入力仕様は論理式で表現される．そのため，入力仕様の学習結果は，形式的仕様記述の論理式で表現できる必要がある．この理由から，本研究では，入力仕様の学習に制約獲得アルゴリズム[8] を適用する．

制約獲得アルゴリズムは，バージョン空間法に基づく学習アルゴリズムである．ありうる制約として述語の集合が前もって定義されているとき，正例と負例からなる訓練例の集合を与えられ，正例に対してのみ真となる述語の連言を獲得する．

Web サービスへの要求メッセージと実行の成否を訓練例として制約獲得アルゴリズムを適用することで，入力仕様に相当する述語の連言が学習結果として得られる．前もって定義しておく述語の集合の各要素について，対応する仕様記述の論理式を用意しておくことにより，得られた学習結果を容易に仕様記述に加えることができる．

以下では，これを実現するための Web サービスの入力仕様のモデル化について説明する．

ある Web サービスが k 個の要素からなる下記の要求メッセージを取るとする．

$$I = \{x_1, ..., x_k\}$$

さらに，この要素を引数に取る，入力仕様を構成するありうる述語の集合を考える．以下の説明では簡単のため，述語には，単項述語と 2 項述語のみがあるとする．i 番目の要素の値を x_i とし，対応する述語を b_i と記述する．また i 番目，j 番目という異なる 2 つの要素の値 x_i, x_j の 2 項述語を $b_{(i,j)}$ と記述する．

単項述語 b_i 及び 2 項述語 $b_{(i,j)}$ は，それぞれ以下のように定義する．

$$b_i : x_i \in \mathit{class} \ \textbf{or} \ x_i \notin \mathit{class}$$

$$b_{i,j} : \{x_i, x_j\} \in \mathit{class1} \times \mathit{class2} \ \textbf{or} \ \{x_i, x_j\} \notin \mathit{class1} \times \mathit{class2}$$

class, *class1*, *class2* などは，Web サービスの要求メッセージの要素が取る値の属するクラスを表す．$x_i \in \mathit{class}$ は，x_i が *class* のインスタンスであることを意味する．$\times$ は直積を意味する．このとき，制約ライブラリ B_s は，あらゆる既知のクラスまたはクラスの組み合わせに関する b_i 及び $b_{i,j}$ を持つとする．

制約獲得アルゴリズムは，論理式 K を考え，正例が与えられたときには，その例が満たさない制約ライブラリの要素の否定を連言として K に加える (初期段階では $K = \mathit{true}$)．負例に対しては，その例が満たさない要素の選言を，K に連言として加える．これによってできる論理式 K を充足させる際に真となる要素の連言が，入力仕様に相当する制約となる．K を充足させるために真となる要素の集合は SAT ソルバによって計算される．

以下で，例を用いて，入力制約の学習の手順について説明する．2 章で述べた機械翻訳 Web サービスに，以下のような要求メッセージが与えられたとする．“ja”, “en”, “ko” はそれぞれ日本語，英語，韓国語を表すコードである．正例はこの Web サービスを実行できた要求メッセージを，負例は実行に失敗した要求メッセージを意味する．

- 正例 $I_1 = \{ja, en, \text{“こんにちは”}\}$
- 負例 $I_2 = \{ja, ko, \text{“Hello”}\}$
- 正例 $I_3 = \{ja, ko, \text{“こんにちは”}\}$

表 1 に，これらの要求メッセージを訓練例としたときの，論理式 K の更新過程を示す．表中の K_i は，I_i による更新結果である．K の各項は制約ライブラリ中に定義された述語である[⋆1]．l_s, l_t, l_e は述語の変数であり，それぞれ翻訳元言語・翻訳先言語・翻訳される文字列の言語を表す．ただし，表記の単純化のため 2 つの言語 l_1, l_2 が同じであることを $l_1 = l_2$ と記した．

表 1 の K_3 では，$(l_t = l_e)$ のみが真となる．そのためこの項が，仕様記述における公理に連言として加えられる．図 1 に示した仕様を更新し，$(l_t = l_e)$ に相当する項を加えたものを**図 4** に示す．`:is` は，ある値がある型の要素であることを調べる組み込みの述語である．`SameLanguagePair` は，2 つの同一の言語コードからなる 2-tuple を表すクラスである．

ただし，K を充足させるために真となる項の集合は，一意に定まるとは限らない．このため，K を充足させる項の連言の集合を仮説集合 H としたとき，仮説 $h_i, h_j \in H$ 間に，一

表 1　制約の学習過程
Table 1　Process of learning input constraints.

K_1	$\neg(l_s \in en) \wedge \neg(l_e \notin ja) \wedge \neg(l_s \in ko) \wedge$ $\neg(l_t \notin en) \wedge \neg(l_t \in ja) \wedge \neg(l_t \in ko) \wedge \neg(l_s \neq l_e)$
K_2	$K_1 \wedge \{(l_s \in en) \vee (l_s \notin ja) \vee (l_s \in ko) \vee$ $(l_t \in en) \vee (l_t \in ja) \vee (l_t \notin ko) \vee \neg(l_s = l_e)\}$
K_3	$K_2 \wedge \neg(l_t \in en) \wedge \neg(l_t \notin ko)$

```
eq domain-translator(e) =
  (1*(e) == english or 1*(e) == japanese)
  and (language-pair(1*(e),
                     get-language(3*(e)))
       :is SameLanguagePair) .
```

図 4　更新された仕様
Fig. 4　An updated axiom.

⋆1 ja, en, ko は言語の集合ではなく単一の言語であるが，ここでは 1 つのインスタンスを持つクラスと見なす．

般性を定義する半順序 $\prec$ を $h_i \prec h_j \equiv (\forall x \in X)[h_i(x) = true \rightarrow h_j(x) = true]$ と定義する．X は，要求メッセージの取り得る値の集合である．$h_i(x) = true$ であるとは，ある仮説 h_i が表す入力仕様の下で，要求メッセージ x で Web サービスが実行可能であることを意味する．ここでの目的は，実行すると失敗することが確実である場合にのみ実行を中止することであるため，$\{h_g \in H | h_g \not\prec (\forall h \in H)\}$ なる h_g 全てを，実行可能性予測のための仮説として記録する．実行可能性予測の際は，全ての仮説 h_g に対応する仕様記述を用いてオペレーション `domain-`*service-name* の簡約を行う．全ての h_g で `false` が得られた場合には，実行不可能であるとして，複合 Web サービスの実行を中止する．

制約ライブラリ中の要素数が大きくなると，制約獲得アルゴリズムによる学習の計算量は増大する．しかし，学習は複合 Web サービス実行の終了または中止後に行うことができるため，複合 Web サービス実行のパフォーマンスを低下させることはない．

4. 実　　験

本研究の提案手法の有効性を確認するため，実際に使用されている複合 Web サービスに対して適用し，無駄な原子 Web サービスの実行がどれほど減少するかを調べた．

ここで対象としたのは，言語グリッド[3] で用いられる，特定ドメインの文章の翻訳を目的とした複合 Web サービス (**図 5**) である．機械翻訳 Web サービスによる翻訳結果に含まれる単語の一部が，専門用語辞書 Web サービスによる翻訳結果と置き換えられる．

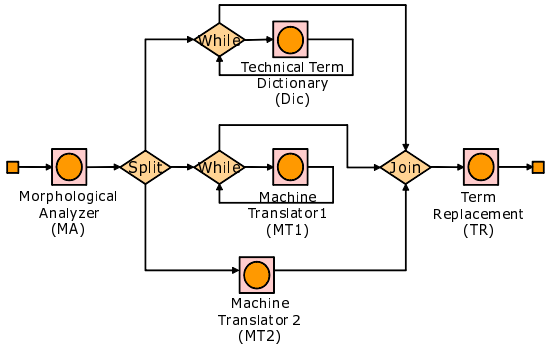


図 5　専門用語翻訳を行う複合 Web サービス
Fig. 5　Composite service for translation in a special domain

各原子 Web サービスの役割と，処理の順は下記の通りである．

(1)　形態素解析により，翻訳される文字列を単語に分割 (Morphological Analyzer)
(2)　下記の並列処理を実行
　(a)　全ての単語を専門用語辞書で翻訳 (Technical Term Dictionary)
　(b)　全ての単語を機械翻訳で翻訳 (Machine Translator 1)
　(c)　翻訳される文字列を機械翻訳で翻訳 (Machine Translator 2)
(3)　Machine Translator 2 で翻訳された文字列に含まれる単語のうち，専門用語辞書で訳語が見つかった単語を置き換え (Term Replacement)

この複合 Web サービス及びその中に含まれる Machine Translator 1，2，Technical Term Dictionary (以下それぞれ MT1, MT2, Dic) は，2 章で述べた機械翻訳 Web サービスと同様の要求/応答メッセージを持つ．Morphological Analyzer (以下 MA) は，解析対象の文とその言語を要求メッセージとして受け取り，解析結果となる単語の配列を応答メッセージとして返す．Term Replacement は，文章と 2 つの単語の配列を要求メッセージとして受け取り，文章中の単語のうち，一方の配列の要素と一致したものを，もう一方の配列の対応する要素に置き換えて応答メッセージとして出力する．これらの処理により，機械翻訳 Web サービスによる翻訳文中の訳語は，可能な場合，専門用語辞書による訳語に置き換えられる．

ここで，MA, MT1, MT2 には，下記の入力仕様を持つ具象 Web サービスを使用した．

- MA：解析対象の言語に，日本語と英語以外が指定されると実行失敗．
- MT1, MT2：指定された翻訳元言語と，入力文の実際の言語が一致しないと実行失敗．また，翻訳される文字列が，100 字以上だと実行失敗．

この入力仕様により，以下のような原子 Web サービスの実行失敗が起こることがある．

- 翻訳される文字列が日本語と英語以外の場合，MA の実行が失敗する．このとき，MA 実行のコストが無駄になる．
- 翻訳される文字列が長い場合，MT2 の実行が失敗する．このとき，MA，MT2 は 1 回，MT1，Dic は反復実行された回数だけ実行コストが無駄になる．
- 日本語文に英単語が含まれるなど，翻訳される文字列が複数の言語の単語を含む場合に，MT1 において，翻訳元言語の指定と与えられる文字列の実際の言語が異なることになり，MT1 の実行が失敗する．MT2 の実行失敗と同様，MA，MT2 は 1 回，MT1，Dic は反復実行された回数だけ実行コストが無駄になる．

以上の条件下で，本研究の提案手法を適用する．各原子 Web サービスの入力仕様は始め未知であり，あらゆる要求メッセージが実行可能と予測されるものとする．また，制約獲得

アルゴリズムの適用のために定義した述語は MA, MT1, MT2 に関するもので，3 章の例で述べた，翻訳元言語・翻訳先言語・翻訳される文字列の言語を規定する述語 (それぞれ日・英・韓の 3 種) に加え，翻訳される文字列の長さを規定する述語 (長・短の 2 種)，及びそれらの組み合わせとした．

さらに，定義した述語に対応するクラスを考え，実験のためあらゆるクラスの組み合わせに相当する要求メッセージを生成した．ここでは翻訳元・翻訳先言語及び翻訳される文字列の言語についてそれぞれ 3 通り，翻訳される文字列の長さについて 2 通りで，計 54 通りを生成して，ランダムな順で与えた．

このときの複合 Web サービスの実行試行回数と，複合 Web サービス・原子 Web サービスの無駄になったのべ実行回数（複合サービスが最後まで実行できなかったときの実行回数）の関係を**図 6** に示す．図 6(a) は無駄になった複合 Web サービスの実行回数について，図 6(b) は無駄になった原子 Web サービスの実行回数について，それぞれ提案手法適用の前後で比較している．

提案手法を適用した場合には，図 6(a)(b) のいずれでも，複合 Web サービスの実行試行回数が増えるに従い無駄になった実行回数の増加が緩やかになっており，仕様の学習が進んでいることがわかる．特に，図 6(b) の原子 Web サービスの実行回数で見ると，図 5 の複合 Web サービスの場合では，反復実行の制御構造による多数の原子 Web サービスの実行が抑止されるため，実行コスト削減の効果は大きい．ただし，無駄な実行回数の増加が完全に止まることはない．これは，MT1 における指定された翻訳元言語と実際の言語の不一致による実行失敗は，MA を実行し，翻訳される文字列の分割結果が明らかになった以後にのみ予測できるためである．

5. 関 連 研 究

これまで，述語論理やペトリネット，有限状態機械などによる複合 Web サービス検証の手法が提案されてきた．以下ではそれらの研究を挙げ，本研究との相違について述べる．

Narayanan らは，OWL-S の前身である DAML-S で記述された複合 Web サービスを対象とし，ペトリネットを用いてモデル化を行った．これにより，制御フローに注目した，目標状態に到達できるかの検証やデッドロックの検出の手法を提案した[5]．

Ankolekar らは，OWL-S をモデルチェッキングのための記述言語に変換し，制御フローとデータフローの不整合などを検出する手法を示した[6]．また Fu らは，同様にモデルチェッキングを用いて，複合 Web サービスにおける Web サービス間のインタラクションプロトコルの検証の手法を提案している[7]．複合 Web サービス記述と，モデルチェッキングのための記述の間の変換に，中間的な表現を介することで，複合 Web サービス記述言語やモデルチェッキングの言語を柔軟に組み合わせられるとしている．

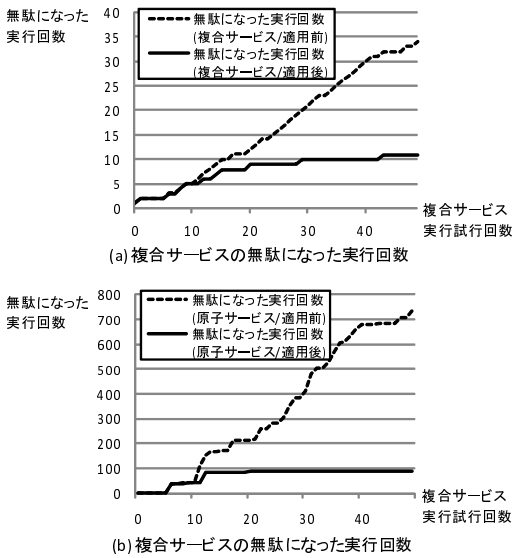


図 6　原子/複合サービスの無駄になった実行回数
Fig. 6　Number of wasteful execution of atomic/composite services

これらの複合 Web サービスの検証を行う従来研究と本研究の相違として，具象原子 Web サービス提供者や抽象複合 Web サービス設計者に関する前提が挙げられる．多くの従来研究では，Web サービスの仕様は完全に明らかであり，また検証の結果原子/複合 Web サービスの仕様に不適切な部分があれば，仕様が修正可能であることを想定している．しかし本研究では，具象原子 Web サービス提供者・抽象複合 Web サービス設計者・アプリケーション開発者がそれぞれ別であることを想定し，不適切な仕様でも修正できない状況を前提とする．そこでは，与えられた要求メッセージごとに実行可能性を予測することが重要となる．

6. お わ り に

本研究では，複合 Web サービスの実行失敗時における，無駄な原子 Web サービスの実

行を抑止する手法を提案した．具象原子 Web サービス提供者・抽象複合 Web サービス設計者・アプリケーション開発者が別であるために，Web サービスの修正が困難であり，また具象原子 Web サービスの完全な仕様が公開されていない状況を想定している．本研究の主な貢献は，以下の通りである．

- 形式的仕様記述による Web サービスのモデル化により，要求メッセージごとの複合 Web サービスの実行可能性予測を実現した．制御構造に基づき原子 Web サービスの仕様記述をボトムアップに合成し，複合 Web サービスの仕様記述を容易に構築できる．
- 制約獲得アルゴリズムを適用し，具象原子 Web サービスの実行を通じて入力仕様を学習することで，実行可能性の予測を改善できることを示した．入力仕様を要求メッセージのクラスに基づいてモデル化し，学習結果を形式的仕様記述による入力仕様に変換可能としている．

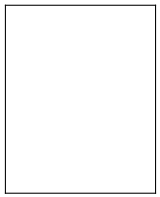
今後の課題として，Web サービスの入出力関係の獲得が挙げられる．入力仕様と同様に，入出力関係の仕様も，具象 Web サービスごとに異なる可能性があり，各具象 Web サービスでの仕様を学習することにより，実行可能性の予測を改善できると考えられる．

謝辞　本研究は，日本学術振興会科学研究費補助金（特別研究員奨励費）及びグローバル COE プログラム「知識循環社会のための情報学教育研究拠点」の補助を受けた．

参 考 文 献

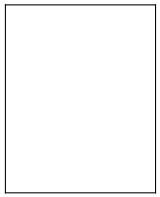
1) : Business Process Execution Language for Web Services (BPEL), Version 1.1., `http://www.ibm.com/developerworks/library/ws-bpel/` (2003).
2) : OWL-S 1.1 Release, `http://www.daml.org/services/owl-s/1.1/` (2004).
3) Ishida, T.: Language Grid: An Infrastructure for Intercultural Collaboration, *IEEE/IPSJ Symposium on Applications and the Internet (SAINT-06)*, pp.96–100 (2006).
4) Hayashi, Y.: Conceptual Framework of an Upper Ontology for Describing Linguistic Services, *the First International Workshop on Intercultural Collaboration*, pp. 246–260 (2007).
5) Narayanan, S. and McIlraith, S.A.: Simulation, verification and automated composition of web services, *the 11th International Conference on World Wide Web (WWW2002)*, pp.77–88 (2002).
6) Ankolekar, A., Paolucci, M. and Sycara, K.: Towards a Formal Verification of OWL-S Process Models, *the 4th International Semantic Web Conference (ISWC2005)*, pp.37–51 (2005).
7) Fu, X., Bultan, T. and Su, J.: Analysis of interacting BPEL web services, *the 13th conference on World Wide Web (WWW2004)*, pp.621–630 (2004).
8) Bessière, C., Coletta, R., O'Sullivan, B. and Paulin, M.: Query-driven Constraint Acquisition, *the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pp.50–55 (2007).
9) Ehrig, H. and Mahr, B.: *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, Berlin (1985).
10) Futatsugi, K. and Nakagawa, A.: An overview of CAFE specification environment-an algebraic approach for creating, verifying, and maintaining formal specifications over networks, *the 1st International Conference on Formal Engineering Methods*, pp.170–181 (1997).
11) Smith, D.R. and Lowry, M.R.: Algorithm Theories and Design Tactics, *Science of Computer Programming*, Vol.14, No.2-3, pp.305–321 (1990).

(平成 20 年 5 月 19 日受付)
(平成 20 年 9 月 11 日採録)



田仲　正弘（学生会員）

2004 年京都大学工学部情報学科卒．2005 年同大学院社会情報学専攻修士課程修了．現在，同大学院社会情報学専攻博士課程に在学中．Web サービス，セマンティック Web，マルチエージェントシステムに興味を持つ．情報処理学会学生会員．



石田　　亨（正会員）

1976 年京都大学工学部情報工学科卒業，1978 年同大学院修士課程修了．同年日本電信電話公社電気通信研究所入所．ミュンヘン工科大学，パリ第六大学，メリーランド大学客員教授など経験．工学博士．IEEE フェロー．情報処理学会フェロー．現在，京都大学大学院情報学研究科社会情報学専攻教授，上海交通大学客員教授．自律エージェントとマルチエージェントシステム，セマンティック Web 技術に取り組む．デジタルシティ，異文化コラボレーションプロジェクトを推進．