1D1-1

階層型計画によるWebサービス分解実行

Web Service Decomposition and Execution by HTN Planning

小出 誠二　　武田 英明
Seiji Koide　　Hideaki Takeda

国立情報学研究所，総研大
National Institute of Informatics, Sokendai

The Web Service Composition is similar but not identical to the task planning, and the Hierarchical Web Service Composition of complex Web Services is similar but not identical to the Hierarchical Task Network (HTN) planning. In addition to the preconditions, the delete lists, and the add lists, we have the inputs and outputs of Web Services. Furthermore, those descriptions include universally quantified typed variables, then we need the typed unification for planning. The complex Web Services have the control construct, which is similar to the one in ordinary programming languages, such as IfThenElse, RepeatUntil, and so on. Therefore, it is difficult to apply the regressive (backward chaining) planning techniques in Web Services. In this paper, we discuss the HTN Web Service composition, decomposition, and execution from the viewpoint of Space State Planning and Axiomatic Semantics. We address the formalization of HTN Web Service Decomposition and Execution.

1. はじめに

複合 Web サービスは複合 Web サービスと実際に実行可能な Web サービスで構成され，エージェントが複合 Web サービスを状況に合わせて解釈することで，実際に実行可能な Web サービス列を実行する．複合 Web サービス合成問題は，タスクが階層的なネットワークを構成している階層的タスクネットワーク (Hierarchical Task Network, HTN) の計画問題に似ている．HTN 計画問題では，階層型計画システムが用いられるが，これまでの古典的な計画システムでは，計画に必要な情報はすべて事前に分かっており，しかも世界は静的であり計画時に変化しないとされ，実行時のことは考慮の範囲外である．セマンティック Web サービスの発見，合成，分解，実行，監視においては，Web 世界は開世界であり不確実であることが前提となっており，その実現は古典的計画問題よりもはるかに困難である．

Sirin ら [Sirin 04] は代表的な階層型計画システム SHOP2 [Nau 03] を Web サービス合成に適応しているが，Web 合成問題は計画問題と似て非なるものであるため [小出 05b]，上記セマンティック Web サービス問題を十分解決したものにはなっていない．不確実な Web 世界を対象とするセマンティック Web サービスにおいては，サービスの発見，合成，分解，実行，監視の各機能は，互いに有機的に結びついており，状況依存エージェントシステムの問題として総合的に考慮され，解決されねばならない [小出 04, 小出 05c]．

本報告では複合 Web サービスを含む Web サービスのネットワークにおいて，Web サービスが同時に抽象・特殊の IS-A 関係により階層的にタキソノミーとして構成されている場合において，サービスの発見，合成，分解，実行，監視を行う手法について考察し，実装のためのアルゴリズムを定式化する．

2. 階層型計画の定式化

2.1 Web サービスと制御構造

○ 単純サービス：　実際に実行可能な Web サービスをここでは単純サービス (SS) とよぶ．単純サービスはグラウンドされる Web サービス (WS), 入力 (In), 出力 (Out), 前提条件 (Pre), 削除リスト (Del)，追加リスト (Add) から構成される．

$$SS = \langle WS, In, Out, Pre, Del, Add \rangle$$

入力はその Web サービスに対する入力，出力はその Web サービスの出力であり，いずれも領域記述 D に定義されたクラスにより型付けされた変数である．前提条件はその単純サービスが実行可能であるための維持条件であり，評価されて真偽あるいは不明を返す条件 $Cond$ をとる．削除リストおよび追加リストの要素と条件 $Cond$ の記述は，領域記述中にあるクラスとインスタンスおよび定数 (リテラル) の連言 (conjunction), 選言 (disjunction), 否定 (negation) を含む一階述語論理記述である．ただし，存在限量子はなく，全称限量子は省略されているものとする．

○ 複合サービス：　複合サービス (CS) は制御構造 (CC) と入力 (In), 出力 (Out), 前提条件 (Pre) から構成される．

$$CS = \langle CC, In, Out, Pre \rangle$$

複合サービスの入力として渡されたインスタンスやリテラルは，(計算されて) 制御構造内部のサブサービスの入力に反映され，内部のサブサービスの出力は (計算されて) その複合サービスの出力に反映される．

○ 制御構造：　ここでは制御構造 (CC) にはプログラム言語 Scheme に相当する制御構造があるものとし，たとえば $IfThenElse$ 節の If 条件には $Cond$ が書かれ，Then 部や Else 部にはサービスや制御構造が書かれるものとする．

制御構造にはプログラムのように，制御の流れが記述されると同時に，上流のサービス出力が下流のサービス入力に反映される情報の流れも記述されることに注意されたい．

2.2 Web サービスの階層型計画問題

Web サービスの計画問題とは，初期状態 S_0,Web サービスのリスト T，領域記述 D が与えられたとき，初期状態 S_0 に対して解釈可能な単純 Web サービスの実行列を求めることである．初期状態 S_0 は単純サービスの解釈によって，削除リストおよび追加リストにしたがって更新を受ける．すなわち，Web

連絡先: 小出誠二，国立情報学研究所総研大，101-8430 東京都千代田区一ツ橋 2-1-2，koide@nii.ac.jp

サービスが選択されたときに状態 S_0 と削除リストにマッチした部分が削除され，追加リストとマッチしてインスタンス化されたものが追加される．ある時点での状態 S_i から更新された状態を S_{i+1} と記述する．ここでは簡単のために，Web サービスを解釈実行するエージェントなるものを考え，エージェントの外界からの必要な情報はすべて状態 S_i に記述されるものとしている．また，単純 Web サービスの実行によって生ずる世界の変化はすべて削除リストと追加リストに不足なく記述されているものとする．

ある条件 $Cond$ が領域記述 D に対して真となるようなモデルを有するとき，その条件を充足可能 (satisfiable) であると言い，充足可能な条件が状態 S_i に対して真となるようなモデルを有するとき，S_i に対して解釈可能 (interpretable) と言うことにする．

あるサービスの前提条件が領域記述 D に対して真となるようなモデルを有するとき，そのサービスを充足可能と言い，充足可能なサービスの前提条件がある状態 S_i に対して真となるようなモデルを有するとき，そのサービスを S_i に対して前解釈可能 (pre-interpretable) と言うことにする．

一般に，あるサービスが充足可能だからと言って，その制御構造が充足可能とは限らないし，あるサービスが前解釈可能だからと言って，その制御構造が解釈可能とは限らない．制御構造が解釈可能とは，その中に含まれる前解釈可能な複合サービスの制御構造が解釈可能であり，解釈可能な単純サービスの列が（計画によって）最低でも一つ得られるということとする．単純サービスは制御構造を含まないため，前解釈可能な単純サービスは解釈可能である．

ある状態 S_i に対して，サービスが前解釈可能でかつその制御構造が状態 S_i に対して解釈可能なとき，そのサービスを解釈可能と言う事にする．解釈可能な制御構造は解釈実行され得る．すなわち，与えられた状態 S_i に対して，制御構造にしたがって T から前解釈可能な Web サービス発見を行い，複合サービスについてはその制御構造について解釈実行し，単純サービスについては削除リストと追加リストにより状態 S_i を S_{i+1} に更新させることができる．素朴な計画においては，ある複合サービスが解釈可能かどうかは，その複合サービスを分解し制御構造を解釈して，そこに含まれる単純サービスを解釈実行した結果でしか分からない．たとえば，$IfThenElse$ 節では If 部の条件が解釈可能であったとき（状態 S_i に対して真となるようなモデルがあったとき）に制御構造の解釈実行として Then 部が解釈可能かどうか試されるが，Then 部にも $IfThenElse$ 節があるような入れ子構造の場合には，内側の $IfThen$ 節において解釈不可能になることは十分考えられることである．

Web サービスの計画問題とは，単純ウェブサービス解釈実行の結果としての状態 S_i の遷移とそれにペアとなっている解釈可能な Web サービスの実行列を求めることである．

Web 世界の開世界性を前提にすると，データは静的で変化しないとしても，計画により解釈実行可能な Web サービス列が本当に実行可能かどうかは分からない．なぜならば，我々の手法では前提条件における解釈不可能なサービスは捨てられるが，$IfThenElse$ 節における If 部条件の解釈不可能な場合に選択される Else 部の解釈には，条件否定の場合もあれば条件不明の場合もあり，Else 部における解釈可能性には不確実さが含まれるからである．

Else 部がない場合には，データが静的であるという前提において，解釈可能な Web サービスは実行可能であるという健全性は確保されるが，解釈不可能だからといってほかに解がないということにはならず，解は不完全である．

2.3 入出力変量 (IO-Fluent)

複合サービスの制御構造において，ある Web サービスの出力が別の Web サービスの入力となるように記述してあったとき，実行時には後段の Web サービスへの入力となるものは前段の Web サービスの出力であるが，計画時にはこれをどのように考えたらよいだろうか．

たとえば，個人の名前を入力してその人の年齢を返すような Web サービスがあったとき，そのサービスの入力には PersonName クラス，出力には PersonAge クラスが定義されているとする．入力インスタンスとして JohnDoe を与えたときの出力は，ただの年齢ではなく，それは JohnDoe の年齢である．これを $Age_{johndoe}$ と表記する．同様に入力インスタンスとして JaneDoe を与えたときの出力は，ただの年齢ではなく，それは JaneDoe の年齢であり，これを $Age_{janedoe}$ と表記する．今性別は問わずに，年齢を入れたときに確定年金の額を知らせる Web サービスがあったとき，$Age_{johndoe}$ の入力に対して得られる年金の額はただの年金額ではなく，JohnDoe の年金額であり，$Age_{janedoe}$ の入力に対して得られる年金の額は JaneDoe の年金額である．すなわち，Web サービスの入出力を考えるとき，それを単なる入出力クラスのインスタンスとしてのみ捉えるのではなく，Web サービスを経るごとに型が変化する fluent として捉えることが必要である．インスタンス in から Web サービス 1,2,3 の順に流れてきた fluent を，$Web3(Web2(Web1(in)))$ のように表現し，エージェントはこの fluent を見れば，それが Web3 の出力クラスのインスタンスであるだけでなく，インスタンス in から Web サービス 1,2,3 の順に流れてきた結果であることが分かるものとする．fluent の意味は最初の入力と，その fluent の履歴によって支えられる．二つの入力から一つの出力となるような場合には，$Web(in1, in2)$ と記述する．

Web サービスの入出力を fluent とすることで，その意味が分かるようになるとして，エージェントは Web サービスの合成を自由にできるようになるのであろうか．たとえば，夫婦の年齢が合計で百歳のときには，特別サービスがあるようなホテル予約 Web サービスを考えたとき，たとえ顧客名簿から Web サービス経由で夫婦関係を調べることができるとしても，エージェントが自分で JohnDoe と JaneDoe が夫婦であるか調べて年齢を合計するようなプログラムを自動生成することは当分の間不可能であろう．しかしそのようなワークフローがあらかじめ抽象的にでも定義されていれば，エージェントが年齢を調べる Web サービスや夫婦関係を調べる Web サービスを発見し，解釈実行することは可能に思われる．エージェントが Web サービスの階層的計画で行うことは，抽象的なワークフローと初期状態から出発して，徐々に抽象ワークフローを特殊化することで解釈実行可能なプログラムを生成することに相当する．

一般に，Web サービスのリスト T の中には制御構造内に含まれるサブ Web サービスについて前解釈可能な Web サービスが複数あることに注意されたい．もし，常に前解釈可能な Web サービスが唯一であるとすればそれは計画というよりも，通常のサブルーチン化された手続き実行に限りなく近いことになる．

2.4 Web サービスの発見

ここでは，ある状態 S_i に対して前解釈可能で，かつ与えられたインスタンス in に対してそのクラスである入力 In を持つような Web サービスを T から発見することを，Web サー

ビスの発見と呼ぶことにする．

状態 S_i にはインスタンスまたはリテラルのみを含む述語論理式が記述してある（すなわち存在限量子も全称限量子もない述語論理式の集合）．前解釈可能な Web サービス発見のために，前提条件 Pre にある条件式 $Cond$ と状態 S_i の各式が比較され，マッチする Web サービスが選択される．状態 S_i の述語論理式中にはインスタンスがあり，条件式 $Cond$ 中にはクラス記述があって，型付きの単一化 [小出 05b] が行われる．

選ばれた前解釈可能な Web サービスのうち，入力インスタンス in に対してそのクラスとなる入力 In が定義してある Web サービスが，計画に採用され得る Web サービスである．

前提条件が無く，入力だけの場合には，もし Web サービスのタキソノミー（IS-A の包含関係）が領域記述 D における入力のタキソノミーと整合的であったなら，Web サービスの発見は簡単である．Web サービスタキソノミーの頂上から探索を始めて，入力インスタンス in に対して最も特殊な概念 (Most Special Concept) であるクラスを入力とする Web サービスを採用すればよい．これを前提条件にまで拡張して，前提条件が単なる述語論理式ではなくクラスとして解釈できれば，領域記述 D における入力および前提条件のタキソノミーと Web サービスのタキソノミーが整合的であるとして，タキソノミーの頂上が与えられてそのサブクラスにおいて最も特殊で適応可能な Web サービスを簡単に見つけることができる．これは DL における計算過程に過ぎない．

一般に計画に採用され得る Web サービスは複数あり，通常の探索問題と同様に，その選択に当たってはどのような探索戦略やアルゴリズムを用いるか，様々な方法が考えられる．

2.5 Web サービスの階層型計画アルゴリズム

状態空間に基づく非常に簡単な Web サービスの階層的計画アルゴリズムの概要を以下に示す．ここで P はプラン，AB は単一化と Web サービスのペアを含むバインディングである．M は Web サービスを表し，$findWebService$ は現在状態 S_i において，入力を考慮して前解釈可能な Web サービスの中から非決定的に 1 個の Web サービスを選ぶ関数，$findBinding$ はインスタンス化途中であるバインディング集合の中から非決定的に一つのバインディングを選ぶ関数である．ここで，バインディングのインスタンス化とは単一化と同義であり，すべての型付けされた変数がインスタンスに単一化されたとき，バインディングがインスタンス化されたと言うことにする．

```
function SWHTNP(S0,T,D)
P = ∅
AB = ∅
S ← S0
makePlan(S,T,D,P,AB)
end

function makePlan(S,T,D,P,AB)
loop
  M ← preinterpretable(S,T)
  if M = ∅ then succeeded return P
  m ← findWebService(S,M)
  if isSimpleService?(m) then
    a ← ⟨m,θ⟩ ; θ は Pre(m) と S の単一化
    b ← ⟨m,ϑ⟩ ; ϑ は In(m) と S の単一化
    AB ← push(⟨a,b⟩,AB)
    bin ← findBinding(last1(P),AB)
    if bin = ∅ then fail
    P = append(P,instantiate(m,bin))
    S = update(S,Del(m),Add(m))
    delete(m,T)
  else if isComplexService?(m) then
    fluit,S,T,D,P,AB =
        interp(controlConstruct(m))
    S = push(fluit,S)
repeat
end

function interp(CC,S,T,D,P,AB)
if CC が変数 then return 変数の値
else if CC =BEGIN then
  loop interp() until 最後の要素の一つ前まで
  return interp( 最後の要素 )
else if CC =SET! then
    変数に interp( 値になるもの ) をセット
else if CC =IF then
  if If 部が解釈可能 then interp(Then 部 )
  else if Else 部がある then interp(Else 部 )
else
  引数 Param_i について順番に
  fluit_i,S,T,D,P,AB
   = interp(Param_i,S,T,D,P,AB) してから,
  fluit = makeFluit(This,fluit1,fluit2,...)
  return fluit,S,T,D,P,AB
end
```

上記 $makePlan$ のアルゴリズムにおいて，いくつかの非決定的選択があるが，ある選択の結果としてプラン作成に失敗したときはバックトラックして，次の非決定的選択肢が選択される．一方，$interp$ のアルゴリズムにおいては，そのような非決定的要素はない．抽象ワークフローがインスタンス化されたプログラムの実行時には，制御構造においてはその制御構造に従ったプログラム実行となり，非決定的要素が入るところはないからである．これは通常の計画問題において，半順序関係が定義されている部分順序計画において，半順序を構成するペアの間に異なるオペレーションが割り込んでもかまわないことと比べると，大きな違いである．

このアルゴリズムにはゴールが現れていないが，制御構造を有する場合に，ゴールから出発して後ろ向きに計画を行うことはほとんど不可能である．ゴールが得られたかどうかの判定は，このアルゴリズムの外側でエージェントによって行われるものと考えている．

3. Web サービスの分解・実行

Web サービスの解釈可能な Web サービス列は，解釈実行系により実際に分解・実行される．解釈実行系として Scheme 仕様の解釈実行系を考えている．以下に [Novig 92] からとった Scheme の解釈実行リスププログラムを示す．

```
(defun interp (x &optional env)
  (cond
    ((symbolp x) (get-var x env))
    ((atom x) x)
    ((case (first x)
       (QUOTE  (second x))
       (BEGIN  (last1
                 (mapcar
                   #'(lambda (y) (interp y env))
                   (rest x))))
       (SET!   (set-var! (second x)
                         (interp (third x) env) env))
       (IF     (if (interp (second x) env)
```

```
            (interp (third x) env)
            (itnerp (fourth x) env)))
(LAMBDA (let ((parms (second x))
              (code (maybe-add 'begin
                               (rest2 x))))
          #'(lambda (&rest args)
              (interp code
                      (extend-env
                        parms args env)))))
(t ;; a procedure application
   (apply (interp (first x) env)
          (mapcar #'(lambda (v) (interp v env))
                  rest x)))))))
```

上記プログラムでは apply となっているところで，関数名あるいは LAMBDA 式が interp で評価され，関数実行されている．Web サービスの制御構造においては，関数名に相当するものはサービス名であり，apply のかわりに複合サービスでは Web サービスの分解実行が，単純サービスでは Web サービスの実際の呼び出しが行われなければならない．複合サービスが正しく Scheme の LAMBDA 式として記述されていれば，それは上記プログラムにおいて正しく分解実行される．あとは Web サービス名が単純サービスであるときに，それにグラウンドされた Web サービスを呼び出すルーチンを追加するだけである．

4. まとめ

本報では状態空間 (State Space) による Web サービスの階層的計画問題の定式化を行いながら，公理論的意味論 (Axiomatic Semantics) の立場に立ったかのような議論を展開したが，それは Web サービスが制御構造を持つためにそうせざるを得なかったからである．すなわち，Web サービスの計画とは解釈可能な単純 Web サービスの実行列を求めることであるが，ある複合 Web サービスが解釈可能かどうかわかるのはそれを解釈実行してはじめて分かるという特徴を持っているためである．通常の階層的計画問題では制御構造はなく，サブタスクの順序関係のみが定義されており，そのことを前提とした計画アルゴリズムがこれまで開発されてきている．SHOP2 [Nau 03] はその代表的なものであるが，通常の計画問題においても，物理的な状態空間 (State Space) をもとにした計画手法から，計画空間 (Plan Space) をもとにした計画手法に主流が移っている．計画空間ベースの計画においては，各ノードは部分的にインスタンス化されたオペレータ集合であり，本報において階層的 Web サービスの計画とは抽象的ワークフローを徐々にインスタンス化して実行可能なプログラムを生成することと主張してきたことと，軌を一にする．さらに公理論的な命題的充足可能技術 (Propositional Satisfiability Techniques) なる計画手法も現れている．公理論的意味論はアルゴリズムの性質を解析するには便利であるが，実装においてはそれを正しく操作的意味論 (Operational Semantics) に落とし込まねばならない．今後は本報の定式化にしたがった Web サービスの階層的計画システムの実装を行う．また，本報では計画時の状態と実行時の状態は同一であることを前提としているが，実世界では世界の状態が変化することを前提に考えざるをえず，我々も応用問題においてそのような問題を抱えているが [小出 05a]，資源制約下での Web サービス計画実行問題についても，今後の課題である．

参考文献

[Nau 03] Nau, D., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., and Yaman, F.: SHOP2: An HTN Planning System, *Journal of Artificial Intelligence Research*, Vol. 20, pp. 379–404 (2003)

[Novig 92] Novig, P.: *Paradigms of Artificial Intelligence Programming*, Morgan Kaufmann (1992)

[Sirin 04] Sirin, E., Parsia, B., Wu, D., Hendler, J., and Nau, D.: HTN Planning for Web Service Composition Using SHOP2, *Journal of Web Semantics*, Vol. 1, pp. 377–396 (2004)

[小出 04] 小出 誠二, 島田 紀一：セマンティック・ウェブサービス用エージェント, セマンティックウェブとオントロジー研究会資料, SIG-SWO-A303-09 (2004)

[小出 05a] 小出 誠二：セマンティック Web サービス用エージェント，ロケット打上運用支援システムへの応用, 人工知能学会誌, Vol. 20, No. 6, pp. 658–665 (2005)

[小出 05b] 小出 誠二：部分順序プラナによるセマンティック Web サービス合成, 人工知能学会セマンティックウェブとオントロジー研究会資料, SIG-SWO-A502-09 (2005)

[小出 05c] 小出 誠二, 島田 紀一：セマンティックウェブサービスのためのタスク処理言語, 人工知能学会セマンティックウェブとオントロジー研究会資料，SIG-SWO-A404-02 (2005)