デザイン問題の計算複雑性

— デザイン問題を解くアルゴリズムは NP 完全である —

COMPUTATIONAL COMPLEXITY OF DESIGN PROBLEM
ALGORITHM FOR DESIGN PROBLEM SOLVING IS NP-COMPLETE藤井 晴行^{*1}

Haruyuki FUJII

It is shown that design can be formulated as satisfiability problem solving, or SAT, if we regard design as a purposive activity to solve a problem. Some characteristics of design computation are discussed on the basis of Cook's Theorem saying that SAT is NP-complete problem and the characteristics of NP-complete problem. It is possible to determine whether a certain design product is actually a solution of a design problem or not in polynomial time. This implies that generate and test is one of pragmatic approaches to solve a design problem if the computational complexity of the algorithm of generating a design product is in the class of polynomial time. It is an important task for design computing study to find such a generation algorithm and to construct a computational model of design thinking where promising design products are created.

Keywords: *Design Science, Problem Solving, Computability, Complexity, NP-Complete*
デザイン科学, 問題解決, 計算可能性, 計算複雑性, NP 完全

1 はじめに

本研究はデザインを新しい仕組みをつくる有目的 (purposive) な活動として捉え、デザインを形式表現する数理論理モデル及び形式言語を構築することを目的としている。デザインを形式表現することにはデザインに関する合理的な議論の基盤を構築するという意義や実際のデザインに用いる計算機システムのアルゴリズムを提供するという有用性がある。

本稿はデザインするということを充足問題^{注1)}を解くこととして定式化できることを示し、充足問題の性質に基づいてデザインすることの特徴を考察する。議論の前提として、デザインはある要求や制約 (提供すべき機能, 堅牢さ, 意匠的美しさなど) を満足する仕組みを, ある制約 (設置環境の諸元, 法律, 予算, 物理法則など) のもとで, 実現するものごと (以下, 人工物) の構成を決定する行為であると仮定する。また, デザインと設計を同義の語として用いる。抽象的にはデザイン・プロセスを要請の記述を人工物の構成に, 制約を満たしつつ, 翻訳するプロセスと捉えることができる。

計算機を利用するデザインは, 現在, 広く普及している。CAD システムを用いて設計図を作成することや, デザインされた人工物のふるまいや性能を計算機シミュレーションによって把握すること

は既に実務レベルで用いられている。また, 複雑な問題を解くための援助として, あるいは, デザインにおける発想のための道具として, 数理科学や人工知能の技法を用いる例も見られる²⁾。これらに関する研究は, 数値計算, 知的 CAD, Design Computation などの観点から取り組まれ, 多くの研究成果が得られている。

一方, デザインすることを数理的, 論理的な問題として定式化した場合, それがどのような性質をもつかを議論する研究は限られている。青木⁵⁾は建築に対する社会的要請を建築の構成に翻訳する設計理論は形式論理的には完全なものとして表現し得ないことを証明したが, これは数少ない研究例の一つであろう。青木⁶⁾は計画行為の妥当性について議論するための集合論的なモデルを提案し, 藤井・青木⁷⁾はそのモデルを踏まえた公理化を行っているが, デザイン問題を解くアルゴリズムの計算複雑性という観点からの考察が完全になされているわけではない。Gero, 中島, 諏訪, 藤井ら⁸⁾⁻¹¹⁾はデザイン・プロセスにおける設計者の状況に依存する思考に着目し, 設計理論の不完全性を詳細に吟味するためのデザイン・プロセスのモデル, デザインの思考過程に関する定性的なモデルを考案している。これらの研究がデザインを内部観測するのに対し, 本研究はデザインを数学的観点から外部観測する。

本研究は日本建築学会2005年度大会にて発表した内容¹⁾を発展させたものである。

^{*1} 東京工業大学大学院建築学専攻 准教授・博士(工学)

Assoc. Prof., Dept. of Architecture, Tokyo Institute of Technology, Dr. Eng.

デザインもしくはその一部を担う計算システムを提案しても、その計算複雑性が現実的でなければ、そのシステムがデザインの実践の用に供することは困難である。本論は、この問題意識から、デザイン・プロセスを問題解決過程として定式化した場合の計算複雑性を考察する。

2 デザイン問題

問題解決の基本は解の候補を生成し、それが実際に解であるか否かを検証するというプロセスを解が見つかるまで繰り返すということである。デザインすることを問題を解決することとして捉えると、デザインのプロセスは、設計案を生成し、それが当の設計に課せられた要求や制約などの要請を満たしているか否かを検証し、検証されればそれを設計解とし、検証されなければ新たな設計案を生成し、検証するという過程を、設計解が見つかるまで繰り返すというプロセスである。本章では、デザインを問題解決とみなすと、デザイン問題は充足問題として定式化できることを示す。

2.1 アルゴリズムの計算複雑性とデザイン

デザインまたはその一部を計算機を利用して行なうためには、デザインを計算 (computation) によって解く問題 (以下、デザイン問題) として定式化し、定式化されたデザイン問題を解く計算機のアルゴリズムをつくる必要がある。また、そのアルゴリズムの実行に必要な時間と記憶容量及びこれらの限界 (以下、計算量) を把握しておくことが重要である。デザイン問題を解くアルゴリズムの計算量が、例えば、スーパーコンピュータ「京」でなければ計算できないほど、非常に膨大であるとしたら、そのアルゴリズムをデザインに利用することは現実的であるとは言えないであろう。

アルゴリズムの計算量は計算時間に注目して議論される。

現実的な望ましいアルゴリズムは多項式時間 (polynomial time) のアルゴリズムである³⁾。多項式時間とは多項式で表わせる計算時間である。多項式時間のアルゴリズムはサイズ n の入力の問題を高々 n の多項式で表わされる計算時間で解くことができる。多項式時間で解かれる全ての決定問題 (decision problem) (集合 $\{F, T\}^*$ から $\{F, T\}$ への写像^{注2)}) の集合を P 問題という。ここで、F は偽 (false) を、T は真 (true) を示す記号である。計算時間の上界のオーダーがサイズ n の入力に対して高々 $O(t(n))$ である問題の全てのものを集合を $Time(t(n))$ とすると P 問題は $Time(t(n^k))$ 問題 ($k = 1 \sim \infty$) の和集合である。デザイン問題を解くアルゴリズムが多項式時間のアルゴリズムであるとき、デザイン問題は計算機によって効率的に解ける問題であるということになる。

P 問題よりも複雑であると考えられている問題のクラスに NP 問題がある。NP は非決定性多項式時間 (Non-deterministic Polynomial Time) の略であり、ここでは数学的に厳密な定義をしないが、ある解の根拠が与えられたとき、その証拠が本当に正しいかどうかを多項式時間で決定できる問題である。非決定論的に多項式時間で解決される問題である。

デザイン問題が NP 問題であるとき、解の候補 (設計案) が、例えば、要請を満たしているというように、本当にデザイン問題の解であるか否かを多項式時間で決定できる問題であるということになる。設計案を生成するアルゴリズムが多項式時間のアルゴリズムであれば、解となる設計案 (設計解) が得られるまでひとつずつ設計

案を生成してそれが本当に解であるか否かを決定するプロセスを繰り返すアルゴリズムも多項式時間のアルゴリズムである。すなわち、デザイン問題を生成検証 (generate and test) の繰り返しによって解く方法は、設計案の生成アルゴリズムが多項式時間のアルゴリズムであれば、現実的な望ましい計算量の方法であるということになる。ただし、最適な設計解を得るためには、最初に得られた設計解を暫定設計解として、暫定設計解と次に得られる設計解とを比較して良い方を暫定設計解とするということをひとつひとつ全ての設計解に関して行い、最終的な暫定設計解を最適設計解とするという手順が必要である。このとき、任意のふたつの設計解のどちらが適当であるかを決定するアルゴリズムが存在することが前提となる。

2.2 設計案の形式表現

設計案を当の設計案がそれを持つか持たないかを判定できるひとつひとつの特徴 (feature) を列挙することによって表わすことにする^{注3)}。例えば、「(設計案は) 住宅である」、「病院である」、「書斎がある」、「外断熱である」、「木造である」など、デザインされるものごとの構成に関する特徴や、「(設計案は) 関連法規を遵守している」、「建設コストが予算内におさまっている」、「省エネルギーである」、「住みやすい」など、デザインされるものごとに対する批評や評価などに関する特徴などがこのような特徴に相当する。前者を構成的特徴、後者を批評的特徴とよび、構成的特徴を設計案それ自体の特徴に対応させ、批評的特徴を設計案の性能や機能に関する特徴に対応させる^{注4)}。

特徴を表記する最小単位の記号の集合を $\Sigma_d = \{d_1, \dots, d_{N_d}\}$ とする。 Σ_d の各記号はひとつひとつの特徴に対応する。 Σ_d は構成的特徴を示す記号の集合 Σ_{ds} と批評的特徴を示す記号の集合 Σ_{dc} の和集合である。ある設計案が持つ構成的特徴を列挙したものは Σ_{ds} の部分集合であり、同じく批評的特徴を列挙したものは Σ_{dc} の部分集合である。設計案をつくるということは要請される批評的特徴 $R_c (\subseteq \Sigma_{dc})$ と構成的特徴 $R_s (\subseteq \Sigma_{ds})$ とを満足する人工物の構成的特徴 $D_s (\subseteq \Sigma_{ds})$ を定めることであると定式化できる。この設計案が持つ特徴は $D_s \cup R_s \cup R_c (\subseteq \Sigma_d)$ の要素である。

設計案が特徴 d_i ($1 \leq i \leq N_d$) を持つか否かを変数 u_i ($1 \leq i \leq N_d$) を用いて表わす。変数の集まりを $U = \{u_1, \dots, u_{N_d}\}$ とする。 u_i は T (true, 真) または F (false, 偽) のいずれかを表す変数 (ブール変数) である。設計案が特徴 d_i を持つことを変数 u_i の値が T であることとして表す。設計案が特徴 d_i を持たないことを変数 u_i の値が F であることとして表す。

変数 u_i の否定を $\overline{u_i}$ と表記する。 u_i の値が T のとき $\overline{u_i}$ の値は F であり、 u_i の値が F のとき $\overline{u_i}$ の値は T である。すなわち、 $\overline{u_i}$ の値が T であることは設計案が特徴 d_i を持たないことを示し、 $\overline{u_i}$ の値が F であることは設計案が特徴 d_i を持つことを示す。

設計案が完成している状態は Σ_d に属する全ての記号 d_i ($1 \leq i \leq N_d$) が示すひとつひとつの特徴について、設計案がそれを持つか否かが確定している状態であると定義する。「書斎がある」か否かが確定していないというように、ひとつでも持つか否かが確定していない設計案は未完成の状態にある。この定義に基づけば、設計案が完成していることはブール変数 U に属する変数 u_i ($1 \leq i \leq N_d$) の値 (T または F) が全て確定していることと同じとなる。

U に属する変数 u_i の値が T, F のどちらの値であるかを示す関

数を U の真理関数 (truth assignment) という。 U の真理関数を $t: U \rightarrow \{T, F\}$ によって示す。 $t(u_i) = T$ は u_i の値が T であることを示す。 設計案が完成しているということは U の真理関数がよく定義されている (well-defined) ことと同じである。

2.3 特徴の関係の形式表現

設計解はそれが本当に解であると確定した設計案である。 要請 $R_c \cup R_s$ を満足するべく定められた構成的特徴を D_s とする設計案が持つ特徴の集合が $D_s \cup R_s \cup R_c (\subseteq \Sigma_d)$ となることは上で議論したとおりである。 これが本当に設計解であることを確かめるには、設計案が集合 $D_s \cup R_s \cup R_c (\subseteq \Sigma_d)$ に属する特徴を全て持つことを確認すればよい。 二つの特徴 d_i と d_j ($i \neq j$) の間には同一の設計案が両方を一緒に持つことがないという関係や、同一の設計案が両方を一緒に持つという関係などがある。 例えば、「大社造りである」という特徴と「超高層である」という特徴と間には、普通、同一の設計案が一緒に持つことはないという関係がある。 また、「超高層である」という特徴と「昇降装置がある」という特徴のように、前者を持つためには後者も持つ必要があるという関係がある。

特徴の間にある関係を変数 $u_i (i \in U)$ を用いて表現するためにリテラルと節を用いる。 変数 u_i とその否定 $\overline{u_i}$ をリテラル (literal) という。 有限個のリテラルの集合を U 上の節 (clause) という。 ある節に属するリテラルのいずれかが T であるとき、その節は充足しているという。 例えば、節 $\{u_1, u_2, \overline{u_3}\}$ は $u_1, u_2, \overline{u_3}$ のいずれかの値が T であれば充足する。

二つのリテラルからなる節には 3 種類の形式がある。 $\{u_i, u_j\}$ なる形式、 $\{\overline{u_i}, \overline{u_j}\}$ なる形式、 $\{u_i, \overline{u_j}\}$ なる形式である。 節はリテラルの集合であるので、 $\{\overline{u_i}, u_j\}$ は $\{u_j, \overline{u_i}\}$ と等しく、 $\{u_i, \overline{u_j}\}$ なる形式と同じである。 これらの節を用いて二つの特徴の間の関係を表わすことにする。 三つ以上の特徴の間の関係もこれらを組合わせて表現可能である。

選言的關係

$\{u_i, u_j\}$ なる形式の節は $t(u_i) = T$ または $t(u_j) = T$ のときに充足する。 この節の充足は $t(u_i) = F$ ならば $t(u_j) = T$ であり、かつ、 $t(u_j) = F$ ならば $t(u_i) = T$ であることと論理的に同じである。 すなわち、この形式の節は変数 u_i, u_j に対応する二つの特徴 d_i, d_j が「設計案が特徴 d_i か特徴 d_j の少なくともどちらか一方を持つ」という選言的關係 (disjunctive relation) にあることを表している。

含意的關係

$\{\overline{u_i}, u_j\}$ なる形式の節は $t(u_i) = F$ または $t(u_j) = T$ のときに充足する。 この節の充足は $t(u_i) = T$ ならば $t(u_j) = T$ であり、かつ、 $t(u_j) = F$ ならば $t(u_i) = F$ であることと論理的に同じである。 すなわち、この形式の節は変数 u_i, u_j に対応する二つの特徴 d_i, d_j が「設計案が特徴 d_i を持つときは特徴 d_j も持つ」、「設計案が特徴 d_j を持たないときは特徴 d_i も持たない」という含意的関係 (implicational relation) にあることを表わしている。

排他的關係

$\{\overline{u_i}, \overline{u_j}\}$ なる形式の節は $t(u_i) = F$ または $t(u_j) = F$ のときに充足する。 $t(u_i) = T$ かつ $t(u_j) = T$ のときは充足しないことと論理的に同じである。 すなわち、この形式の節は変数 u_i, u_j に対応する二つの特徴 d_i, d_j が「設計案は特徴 d_i と特徴 d_j を同時に持た

ない」という排他的関係 (exclusive relation) にあることを表わしている。

2.4 デザイン問題の定式化

設計案が満足すべき要請を節の集合の充足を用いて表わす。 C を $U = \{u_1, \dots, u_{N_d}\}$ 上の節の有限集合とする。 C はそこに属する全ての節が充足するとき、そのときに限り、充足するものとする。 例えば、節の有限集合 $C = \{c_1, c_2, c_3\}$ は節 c_1, c_2, c_3 の全てが充足するとき、そのときに限り、充足する。

設計案への要請がその設計案が持たなくてはならない特徴を示すものであれば、その特徴に対応する変数ひとつからなる節を用いればよい。 節 $\{c_i\}$ の充足は「設計案が特徴 d_i をもつ」ことに対応する。 また、特徴間の関係に関する要請は上に示した 3 種類の形式の節の組合せによって表わされる。

U 上の節の有限集合 C は設計案に対する要請を表わすので、要請を満足する設計解を見つけるデザイン問題は C を充足する真理関数を求める問題として定式化される。 C を充足する真理関数をみつけることは、 C が表わす要請を満足する設計案、すなわち設計解をみつけることに等しい。 C を充足する真理関数がひとつもないこと、すなわち、全ての真理関数が C を充足しないことがわかれば、 C が表わす要請を満足する設計解が存在しないことが確認されたことになる。

U 上の節の有限集合 C を充足する真理関数が存在するか否かを決定する問題は充足問題である^{3), 4)}。 本章のここまでの議論はデザイン問題を充足問題として定式化できることを示すものである。

充足問題として定式化したデザイン問題は解 (問題を充足する真理関数) が必ずしもひとつに定まるわけではないという点で、複数の解がありうるというデザインに関する直感的認識と整合的である。 また、本研究の定式化は、デザイン問題を解くプロセスの特徴 (例えば、解が暫定的な解候補に依存するか) には言及していない。

デザイン問題の例

特徴を表記する最小単位の記号の集合を $\Sigma_{d6} = \{d_1, \dots, d_6\}$ とする。 $U_6 = \{u_1, \dots, u_6\}$ とし、変数 u_i の値が T であることは設計案が特徴 d_i を持つことを表わすものとする。 このとき、以下の要請を満足する設計解を求めるデザイン問題は下式を充足する U の真理関数を求める充足問題となる。

要請 1: 設計案は特徴 d_5 を持つ。

要請 2: 設計案は特徴 d_1 を持つならば特徴 d_4 を持つ。

要請 3: 設計案は特徴 d_1 と特徴 d_2 をもつならば特徴 d_5 を持つ。

要請 4: 設計案は特徴 d_4 を持たずかつ特徴 d_3 と特徴 d_2 を持つならば特徴 d_5 を持つ。

$$\{\{u_5\}, \{\overline{u_1}, u_4\}, \{\overline{u_1}, \overline{u_2}, u_5\}, \{u_4, \overline{u_3}, \overline{u_2}, u_5\}\} \quad (1)$$

例えば、特徴 d_1 を持つ ($t(u_1) = T$)、特徴 d_2 を持つ ($t(u_2) = T$)、特徴 d_3 を持つ ($t(u_3) = T$)、特徴 d_4 を持つ ($t(u_4) = T$)、特徴 d_5 を持つ ($t(u_5) = T$)、特徴 d_6 を持つ ($t(u_6) = T$) 設計案は設計解のひとつである。

3 NP 完全問題としてのデザイン問題の計算複雑性

充足問題は NP 完全問題であるということが Cook の定理^{注5)}によって示されている^{3), 4)}。 要請を満足する設計解を見つけるデザ

イン問題は充足問題として定式化できることを節 2.4 で示した。問題解決過程として定式化したデザインの問題が NP 完全問題であることを確認したわけである。

Cook の定理

充足問題は NP 完全問題である。

■

NP 完全問題の計算複雑性から見たデザイン問題の性質を考察する。

NP 完全問題の定義

問題 L が NP 完全問題であるということは次の条件が満たされることである。

1. 問題 L はクラス NP に属する問題である。
2. 問題 L はクラス NP に属する全ての問題 L' から多項式時間変換可能な問題である。

■

問題 L' から問題 L に多項式時間変換可能であるとは、問題 L' の各入力 x を問題 L の入力例 $f(x)$ への変換 f を計算する多項式時間のアルゴリズムが存在し、問題 L' の各入力 x に対する出力と問題 L の入力 $f(x)$ に対する出力が全て同じであるということである。

NP 完全問題の定義は次の補題と同じである。

補題

問題 L と問題 L' がクラス NP に属する問題であり、問題 L' が NP 完全問題であり、かつ、問題 L' から問題 L に多項式時間変換可能であるとき、問題 L も NP 完全問題である。

■

NP 完全問題の性質は全てが明らかになっているわけではない。ほとんど多くの専門家は $NP \neq P$ であると信じている。しかし、 $NP \neq P$ は証明されているわけではない。ある問題 L が NP 完全問題であるとしてもクラス P に属する可能性があることは否定されていない。問題 L がクラス P に属するならば $NP = P$ となる。ただし、問題 L は NP に属する問題の中で多項式時間のアルゴリズムを見いだすのが最も難しい問題でもある³⁾。

これらのことから、デザイン問題を解く多項式時間のアルゴリズムを見いだすことは容易ではないということがわかる。しかし、設計案が本当にデザイン問題の解であるか否かは多項式時間で決定できるので、設計解が得られるまでひとつずつ設計案を生成してそれが本当に解であるか否かを決定することを繰り返すアルゴリズムは多項式時間で計算できる。すなわち、デザイン問題を生成検証の繰り返しによって解く方法が、デザイン問題を解く多項式時間のアルゴリズムが見いだされるまでは、現実的な計算量の方法である。設計解であると確定できる設計案を繰り返し回数が少ない段階で生成できるほど計算時間は短くなる。そのような設計案の生成方法を考案することがデザイン問題を解く実効的なシステムの構築の鍵である。人間であるデザイナーは、実際、ヒューリスティクスを用いて有望な設計案を生成することによって限られた時間の中で設計解を見いだしている。このようなヒューリスティクスを計算するアルゴリズムを見つけることはデザインに計算機を利用するための一助となる。

デザイン問題の計算複雑性は問題のサイズが大きくなるにつれて非常に増加する可能性がある。多項式時間のアルゴリズムの計算の

オーダーは $O(t(n^k))$ ($k = 1 \sim \infty$) である。 k の値が大きい場合、 n の増加が計算時間の増加に大きく影響する。デザイン問題を解くプログラムが入力サイズの小さいときには現実的な計算量であるように見えても、実用に供する入力のサイズでは膨大な計算時間を要する可能性がある。デザイン問題の計算的解法の有効性を正当に評価するためには k の大きさの範囲を、大まかにでも、デザイン問題の計算的特徴に基づいて、おさえておくことが肝要である。そのために、デザイン問題の解決に関わる計算や操作の性質を詳細に分析し、計算量のオーダーが既知の計算や操作（例えば、デザイン解の候補を評価値の順に並べる操作の計算量のオーダーは最悪で $O(t(n^2))$ であることが知られている）の組み合わせとしてデザイン問題を構成し、問題全体の計算量のオーダーを推定する方法がある。これはデザイン問題の計算複雑性を明らかにするために残されている課題である。

デザイン問題の計算複雑性と計算的解法に関してさらに考慮しなくてはならない性質がある。実際に解くべき問題が不良定義問題 (ill-defined problem) であるということである。デザイン問題の計算複雑性に関するここまでの議論はデザイン問題がよく定義されていることを前提としている。設計案に対する要請は、しばしば、デザインの進行中に変化する。設計案を吟味しているときに新たな要請が生じることはよくある。このような要請を予め問題の定義に加えておくことはできない。また、設計案の特徴に対応する変数がデザインの進行中に変化する場合もある。新たな要請が生じるときと同様に、設計案を吟味しているときに考慮すべき変数が増えることがしばしばある。これらの性質を考慮してデザイン問題を定式化することも重要な課題である。定式化のしかたによってはデザイン問題が NP 完全問題よりも複雑なクラスに属するかもしれない。そうであるか否かを確認することもデザイン問題の計算複雑性に関する課題である。

4 まとめ

デザインを問題解決として捉えるとデザイン問題は充足問題として定式化できることを示し、充足問題が NP 完全問題であることを踏まえてデザイン問題の性質を考察した。

謝辞

本研究は文部科学省科学研究補助金（基盤研究・課題番号 13650667, 同 22615014）を受けた研究を発展させたものである。

参考文献

- 1) 藤井晴行: 充足問題を解く行為としてのデザイン, 日本建築学会大会学術講演梗概集, Vol. E-1, pp. 593-594, 2005.
- 2) 日本建築学会編: 建築のデザイン科学, 京都大学学術出版会, 2012.
- 3) 竹内外史: P と NP - 計算量の根本問題, 日本評論社, 1996.
- 4) Gary, M. R. and Johnson, D. S.: *Computers and Intractability - A Guide to the Theory of NP-Completeness*, W. H. Freeman and Co., 1979.
- 5) 青木義次: 計画の構造と手法, 建築研究報告 第 80 号, 建設省建築研究所, 1977.6.
- 6) 青木義次: 計画行為の妥当性に関する論証, 日本建築学会計画系論文集, No.527, pp.143-148, 2000.1.
- 7) 藤井晴行, 青木義次: 計画行為の妥当性論証にむけた公理化, 日本建築学会計画系論文集, No.532, pp.157-162, 2000.6.
- 8) Gero, John S., Fujii, H.: A Computational Framework for Concept Formation in a Situated Design Agent, *Knowledge-Based Systems*,

Vol.13, No.6, pp.361-368, 2000.11.

- 9) 藤井晴行, 谷本潤: 意図的とみなせる行動の発生に関する思考実験, 日本建築学会計画系論文集, No.547, pp.149-155, 2001.9.
- 10) 藤井晴行, 中島秀之, 諏訪正樹: 構成論的方法論から見たイノベーションの諸相—建築を題材として. 情報処理学会誌. Vol. 49. No. 4. pp.1571-1580, 2008.4.
- 11) 藤井晴行, 中島秀之: デザインという行為のデザイン, 認知科学, Vol.17, No.3, pp.403-416, 2010.9.

注

- 注 1) 充足 (可能性) 問題 (satisfiability problem, SAT) とはブール変数 (Boolean variable, 真 (true) または偽 (false) のいずれかを値とする変数) と論理演算 (「～でない」, 「～または～」, 「～かつ～」など, ただし, 「～」はブール変数またはブール式を示す) によって構成されるブール式 (Boolean expression) の値を真とするようにブール式に含まれるブール変数の真偽をうまく定めることができるか否かを判定する問題である.
- 注 2) A が集合を示すとき A^* は 0 個以上の任意の個数の A の直積集合 ($A \times \cdots \times A$) の和を示す.
- 注 3) 特徴 (feature) は互いに独立であっても依存関係にあってもかまわない. 特

徴間の依存関係は 2.3 節で説明する形式表現を用いて定義することが可能である. 依存関係の定義に出現しない特徴は他の特徴から独立している.

- 注 4) 形式論理を用いて設計案を詳細に記述する場合, 「個体 c は P なる性質を持つ」, 「個体 c と個体 d は R なる関係を持つ」などの設計案の特徴を示す文を, 個体 (c, d) を主部に置き, 性質 (P) や関係 (R) に置いて, それぞれ, $P(c)$, $R(c, d)$ と表記するような述語論理の言語を, 一般的に, 用いる. 本研究では, デザイン問題の計算複雑性の検討に適した定式化のため, 「(設計案は) 住宅である」, 「(設計案は) 住みやすい」など特徴を命題として扱い, その真偽に注目している.
- 注 5) Cook の定理は, 1) 充足問題が NP 問題であること, 2) NP 問題である全ての問題が多項式時間で充足問題に変換できることを示すことによって, 充足問題が NP 完全問題であることを証明している (例えば, 文献 2, pp.11-15). 充足問題が NP 完全問題であることから, 充足問題から多項式時間で変換できる問題は NP 完全問題であることが示される. 充足問題を充足問題以外の問題として定式化したデザイン問題に多項式時間で変換することによってもデザイン問題が NP 完全問題であることが示される.

(2013年4月4日原稿受理, 2013年9月2日採用決定)