PCP定理とその証明

清水 伸高 (東京科学大学)

Contents

序文

このノートは, 計算量理論で 90 年代に証明された重要な結果である PCP 定理とその証明についての講義ノートである. 計算量 (computational complexity) とは, 問題を解くために必要な計算リソースの量 (例えば計算時間, 記憶領域のサイズ, 乱択や量子性の有無や量) を意味し, 計算量理論 (computational complexity theory) とはそれぞれの問題の計算量を明らかにするための理論である. PCP 定理とは, 判定問題 (Yes か No で答える問題) の検証に要する計算量に関する結果であり, 端的に言うと, ある命題が真であると主張する証明が文字列として与えられたとき, その証明を検証するためには, 通常, 全ての文字を見て確認する必要があるが, PCP 定理によれば, その証明の一部だけを見ることで, その命題が真であるか否かを確率的に検証することができるという驚くべき結果である. 例えば, ある実行列 A と実ベクトル b に対して線形方程式系 $Ax = b$ は解を持つ, という命題を考えてみよう. この命題が真であるならば実際に解の一つ x を証明として提示することができるが, その証明が正しいかどうかを検証するためには検証者は Ax を実際に計算し, その各成分が b と一致するかを確認する必要がある. ところが PCP 定理によれば, 巧妙に構成された証明 π を提示することにより, その証明 π 全ての文字を見ることなく, 99%の確率で正しく検証できるのである (ここでは確率的な検証, つまり検証者はランダムネスを用いた検証を行う設定を考える).

このように, 局所的な情報だけを使って全体の構造を推測できるという PCP 定理の性質は, 単に理論的に興味深いだけでなく, 誤り訂正符号の構成, 確率論的手法の脱乱択化, 最適化問題の近似率限界の導出など, 理論計算機科学において広大な応用を持つ.

Chapter 1

導入

この集中講義では PCP 定理と呼ばれる計算量理論の基本的な結果について解説し, その証明を与える. PCP 定理は 1998 年に Arora and Safra [AS98] and Arora, Lund, Motwani, Sudan, and Szegedy [ALMSS98] によって証明された. この証明は代数的な手法に基づく誤り訂正符号を技巧的に組合せたものであり, 難解なものであったが, その後 Dinur [Din07] によってより簡潔な証明が与えられた. この講義では Dinur [Din07] による比較的簡単な証明を紹介する. ちなみに Dinur はのちにこの業績によりゲーデル賞を受賞している.

1.1 計算量理論の復習

まずは計算量理論のどの教科書にも載っているような基礎的な用語の定義を与える. これらの用語に明るい読者はセクション 1.2.2 から読み始めても良い. なお, このノートではアルゴリズムの定義 (チューリング機械の定義) は省略し, アルゴリズムについて述べる際は具体的な計算の手続きを述べる[1].

まずは基本的な記号の定義を与える:

- オーダー記法: 二つの関数 $f, g\colon \mathbb{N} \to \mathbb{N}$ に対し, $f(n) = O(g(n))$ であるとは, ある定数 $c > 0$ が存在して, 十分大きな全ての $n \in \mathbb{N}$ に対して $f(n) \leq cg(n)$ が成り立つことをいう. また, $f(n) = \Omega(g(n)), f(n) = o(g(n)), f(n) = \omega(g(n))$ なども同様に ($n \to \infty$ として) 定義する.

- 自然数 $n \in \mathbb{N}$ に対して $[n] = \{1, \ldots, n\}$ とする.

- 有限集合 S に対し, $x \sim S$ と書いたとき, x は S から一様ランダムに選ばれた元であることを意味する.

- 集合 S 上のベクトル $x \in S^n$ および $I \subseteq [n]$ に対し, $x_I \in S^I$ を, x の I への制限, すなわち, $x_I(i) = x(i)$ $(i \in I)$ と定義する.

[1] ひとまず Python や C 言語などで実装されたプログラムを考えれば良い. ただし, 計算機内では全ての数値は有限桁の二進数で表記されており, その読み書きや演算には少なくとも桁数に比例した計算時間がかかる. なお, $\sqrt{2}$ といった無理数は本来は有限桁で打ち切った近似値を扱うが, そのような小数はこの講義では扱わず, 特に断りのない限りは整数値のみを考える. また, 記憶領域へのアクセスは定数時間で行えると仮定する (チューリング機械であればテープの移動にかかる時間も考慮する).

- $\{0,1\}^* = \bigcup_{n\in\mathbb{N}}\{0,1\}^n$ を有限長の二進文字列全体とする.
- $x \in \{0,1\}^*$ に対して $|x|$ を x の文字数とする.
- アルゴリズム A に対し, $A(x)$ を入力 $x \in \{0,1\}^*$ に対するアルゴリズム A の出力とする. ここで, 単に「アルゴリズム」と言った場合は決定的アルゴリズムを指し, 乱択アルゴリズムについては明示的に言及する.
- 関数 $T(n)\colon \mathbb{N} \to \mathbb{N}$ を考える. 十分大きな全ての $n \in \mathbb{N}$ と全ての $x \in \{0,1\}^n$ に対して, A が $A(x)$ を出力するまでにかかる計算ステップ数の最大値が高々 $T(n)$ であるとき, アルゴリズム A の計算量は $T(n)$ であるという. 特に, ある (n に依らない) 定数 $c > 0$ が存在して計算量が $O(n^c)$ で抑えられるアルゴリズムを**多項式時間アルゴリズム**という[2]
- 慣例的な記法だが, 二つの文字列 $x, y \in \{0,1\}^*$ を入力として受け取るアルゴリズムは $A(x,y)$ と表す. 三つ以上の場合も $A(x,y,z)$ などと表す.

計算量理論で最も基本的な問題群として判定問題と呼ばれる問題群がある.

定義 1.1.1 (判定問題)

部分集合 $L \subseteq \{0,1\}^*$ を**判定問題 (または言語)** といい, アルゴリズムに与える L の入力を**インスタンス**という. また, インスタンス $x \in \{0,1\}^*$ は $x \in L$ であるとき, 判定問題 L の **Yes インスタンス**といい, そうでない場合は **No インスタンス**という.
文字列 $x \in \{0,1\}^*$ に対し $L(x) \in \{0,1\}$ を, $x \in L$ かどうかの指示関数, すなわち $x \in L$ ならば $L(x) = 1$, そうでなければ $L(x) = 0$ と定義する.
アルゴリズム A は, 任意の $x \in \{0,1\}^*$ に対して $A(x) = L(x)$ が成り立つとき, A は L を解くという.

この講義では「効率的に解ける」といった場合, 多項式時間アルゴリズムによって解けることを指す. そのような判定問題の集合をクラス P という.

定義 1.1.2 (クラス P)

判定問題 L は, それを解く多項式時間アルゴリズムが存在するとき, P に属するという.

次に乱択アルゴリズムについて定義する. 端的に言えばアルゴリズムの内部でコイントスを行うものを乱択アルゴリズムという. ここでは明示的にランダムシードを受け取るアルゴリズムを乱択アルゴリズムと呼ぶことにする.

定義 1.1.3 (乱択アルゴリズム)

入力 $x \in \{0,1\}^*$ とは別にランダムシード (乱数表) と呼ばれる別の文字列 $s \in \{0,1\}^*$ を受け取るアルゴリズムを**乱択アルゴリズム**といい, ランダムシードであることを強調するために $A(x;s)$ などと表す. なお, 任意の $x, s \in \{0,1\}^*$ に対して $A(x;s)$ は有限

[2]計算モデルによって一つ一つの計算ステップの定義は異なるが, 原理的には 1bit の演算や記憶領域への読み書きの回数と思えばよい. 多項式時間で動くかどうかの議論であれば, 多くの古典的な計算モデルは等価である.

時間で停止するとし, その計算量は $|x|$ のみに依存する関数で表せるとする. このとき, ランダムシード s の長さを常に A の計算量で上から抑える. すなわち, A の計算量が $T(n)$ であるとき, 十分大きな全ての $n \in \mathbb{N}$ と全ての $x \in \{0,1\}^n$ に対して A が読み込む s の文字数は高々 $T(n)$ であるため, $s \in \{0,1\}^{T(n)}$ であると仮定する. しばし, ランダムシード s を明記する必要が特にない場合は $A(x)$ と表す.
乱択アルゴリズムのランダムシードに関する確率, 期待値, 分散を議論する際は記号として $\Pr_A[\cdot], \mathbb{E}_A[\cdot], \mathbf{Var}_A[\cdot]$ を用いる. 乱択アルゴリズム A が判定問題 L を解くとは,

$$\Pr_A[A(x) = L(x)] \geq 2/3$$

が成り立つことをいう.

また, 入力とは別に文字列へのオラクルアクセスを受け取るアルゴリズムを考える.

定義 1.1.4 （オラクルアルゴリズム）

文字列 $\pi \in \{0,1\}^*$ に対し, π への**オラクルアクセス**を持つアルゴリズム $A^\pi(x)$ とは, 計算途中で π の指定された位置の文字を読むことができるアルゴリズムである. すなわち, π の i 番目の文字を読む操作を $A^\pi(x)$ の計算過程中に $O(\log |\pi|)$ 時間で行うことができるアルゴリズムである.[a] 同様に乱択オラクルアルゴリズムについても定義できる.

[a] 自然数 $i \in [|\pi|]$ を指定するために $O(\log |\pi|)$ ビットを定めなければならないため, $O(\log |\pi|)$ 時間を仮定している.

1.2 検証の計算量

数学全般における検証とは, ある命題が真であると主張する証明が与えられたとき, その証明が実際にその命題を正しく証明しているかどうかを確認することを意味する. 論文や記述試験の証明の査読や採点をイメージしてもらうとわかりやすいだろう. 計算量理論では検証やその計算量の議論は重要な研究テーマであり, その検証に要する計算量が議論される.

1.2.1 効率的な検証とクラス NP

判定問題 L と入力 $x \in \{0,1\}^*$ を与えられたとき, $x \in L$ かどうかを審議したい. ここで $x \in L$ を主張する証明が文字列 $\pi \in \{0,1\}^*$ で与えられたとする. このとき, **検証者**と呼ばれるアルゴリズムは x と π を読み込んで $x \in L$ かどうかを判定する. この判定を多項式時間で行えるとき, その判定問題 L の集合を NP という.

定義 1.2.1 （クラス NP）

判定問題 L は, 以下を満たす多項式時間アルゴリズム V と多項式 $p\colon \mathbb{N} \to \mathbb{N}$ が存在するとき, L は NP に属するという: アルゴリズム V は入力として $x, \pi \in \{0,1\}^*$ を受け取り, 0 または 1 を出力する.

1. もし $x \in L$ ならば, ある $\pi \in \{0,1\}^{p(|x|)}$ が存在して $V(x,\pi) = 1$ となる.

2. もし $x \notin L$ ならば, 全ての $\pi \in \{0,1\}^{p(|x|)}$ に対して $V(x,\pi) = 0$ となる.

また, このようなアルゴリズム V を **NP 検証者**といい, π を **NP 証拠**という.

注釈 1.2.2 （クラス P と NP の関係）

判定問題 L が P に属するならば $L \in \mathsf{NP}$ である. 実際, 受け取った $x \in \{0,1\}^*$ に対して $L(x)$ を計算してそれを出力する検証者を考えばよい. すなわち $\mathsf{P} \subseteq \mathsf{NP}$ である. 一方, 逆側の包含関係 $\mathsf{NP} \subseteq \mathsf{P}$ が成り立つかどうかは P vs NP 問題と呼ばれる計算量理論における最も重要な未解決問題であり, 多くの研究者は $\mathsf{NP} \subseteq \mathsf{P}$ が成り立たないと信じている.

例 1.2.3 （合成数判定問題）

判定問題 $L = \{x \in \{0,1\}^* : x \text{ は合成数 }\}$ を考える. このとき, 検証者は x と π を読み込んで, $\pi \notin \{1, x\}$ かつ π が x を割り切るかどうかを判定する. もしも $x \in L$ である場合, 合成数なので非自明な約数を証拠 π として与えれば $V(x,\pi) = 1$ となる. そうでない場合, 非自明な約数は存在しないため必ず $V(x,\pi) = 0$ となる. このアルゴリズム V は入力長 (つまり数値の二進表現したときのビット長) に関する多項式時間で動作するため, L は NP に属する.

例 1.2.4 （グラフ彩色問題）

自然数 $k \geq 2$ とグラフ $G = (V, E)$ に対し, 関数 $c\colon V \to [k]$ が全ての辺 $\{u, v\} \in E$ に対して $c(u) \neq c(v)$ を満たすとき, c を G の k-彩色といい, k-彩色が存在するようなグラフは k-彩色可能であるという. 任意の $k \geq 2$ に対し, 判定問題

$$L = \{G \in \{0,1\}^* : G \text{ は } k\text{-彩色可能 }\}$$

は NP に属する. 検証者は G と π を読み込んで, π が G の k-彩色であるかどうかを判定する. もしも $G \in L$ である場合, G は k-彩色可能であるため, k-彩色の証拠 π を与えれば $V(G,\pi) = 1$ となる. そうでない場合, G は k-彩色可能でないため必ず $V(G,\pi) = 0$ となる. このアルゴリズム V は多項式時間で動作するため, L は NP に属する.

演習問題 1 （素数判定問題）

自然数 $n \in \mathbb{N}$ に対し, 判定問題 $\mathrm{PRIMES} = \{a \in \mathbb{N} : a \text{ は素数 }\}$ を考える. この問題は P に属することが知られている [AKS04] が, その複雑なアルゴリズムを用いずに初等的に $\mathrm{PRIMES} \in \mathsf{NP}$ を示したい. そのために, 以下の事実を用いる:
任意の自然数 $a \in \mathbb{N}$ と $\gamma \in \{1, \ldots, a-1\}$ に対し, $\gamma^0, \gamma^1, \ldots \pmod a$ は周期的である. さらに, 以下が成り立つ:

- a が素数であるならば, ある $\gamma \in \{1, \ldots, a-1\}$ が存在して $\gamma^0, \gamma^1, \ldots \pmod a$ の周期が $a-1$ である[a].

- 一方, a が素数でないならば, 全ての $\gamma \in \{1, \ldots, a-1\}$ に対して $\gamma^0, \gamma^1, \ldots \pmod a$ の周期は $a-1$ 未満である. 特に, その周期 L は $a-1$ を割り切る.

これらの事実を用いて, 以下の小問に答えよ.

1. 次の検証者 V_1 を考える: 入力 $a \in \mathbb{N}$ と証拠 $\gamma \in \{1, \ldots, a-1\}$ に対し, $\gamma^0, \gamma^1, \ldots, \gamma^{a-2} \pmod a$ を全て検証し, これら全て相異なるかどうかを判定する. この検証者 V_1 が多項式時間アルゴリズム**でない**理由を簡潔に説明せよ.

2. 入力 $a \in \mathbb{N}$ に対し, a が素数であることの証拠として, 原始元[a] $\gamma \in \{1, \ldots, a-1\}$ および $a-1$ の素因数分解 $a-1 = p_1^{\alpha_1} \cdots p_k^{\alpha_k}$ および各 p_i が素数であることの証拠を再帰的に与える. この証拠を用いて, 検証者 V_2 は a が素数であるかどうかを多項式時間で判定できることを示せ.

[a] このような γ を原始元という.

1.2.2 局所的な検証とクラス PCP

一般に $x \in L$ かどうかの検証では, 証明 π の全ての文字を読む必要がある. しかし, 証明 π のうちの一部の文字を読むだけで $x \in L$ かどうかを**確率的に**判定できる場合がある. そのような性質を持つ証拠を**確率的検証可能な証拠** (Probabilistically Checkable Proof, PCP) という.

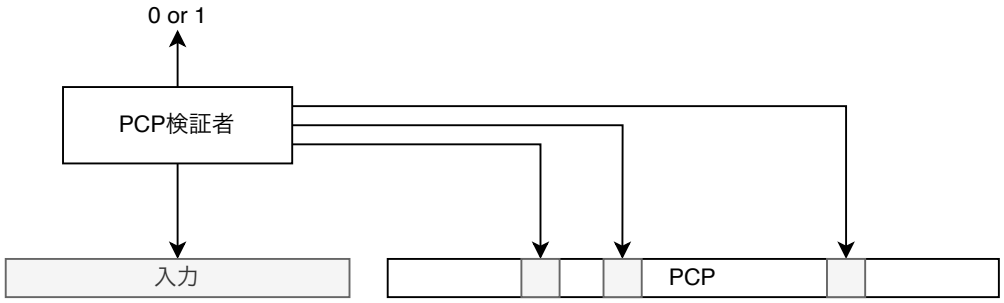


図 1.1: 確率的検証可能な証拠の概念図. 検証者は乱数を用いて証拠 π のうちの一部の文字のみを読み, それに基づいて判定を行う.

定義 1.2.5 (確率的検証可能な証拠)

二つの関数 $r, q \colon \mathbb{N} \to \mathbb{N}$ に対し, $\mathsf{PCP}(r, q)$ を以下の性質を持つ判定集合 L の集合とする: ある多項式時間オラクル乱択アルゴリズム V が存在して, 任意の $x \in \{0,1\}^*$ に対し,

1. もし $x \in L$ ならば, ある $\pi \in \{0,1\}^*$ が存在して, (V の乱択に関して) 確率 1 で $V^\pi(x) = 1$ となる (**完全性**).

2. もし $x \notin L$ ならば, 全ての $\pi \in \{0,1\}^*$ に対して, (V の乱択に関して) 確率 $1/3$ 以上で $V^\pi(x) = 0$ となる (**健全性**).

3. さらに, 入力長が $n = |x|$ のとき, $V^{\pi}(x)$ はオラクル π のうち高々 $q(n)$ 個の文字を読み, そのランダムシード長は $r(n)$ で抑えられる.

このようなオラクル乱択アルゴリズム V を **PCP 検証者**といい, 証拠 π を **PCP** という.

注釈 1.2.6 （PCP の長さ）

$\mathsf{PCP}(r, q)$ の証拠 π の長さは $q(n)2^{r(n)}$ で抑えられる. 各ランダムシード $s \in \{0,1\}^{r(n)}$ に対して検証者は π のうち高々 $q(n)$ 個の文字を読むため, 全てのランダムシードを列挙すると, アクセスされる可能性のある π の文字数は高々 $q(n)2^{r(n)}$ で抑えられる.

一般にランダムシード長 $r(n)$ と読み込む文字数 $q(n)$ が小さいほど良い PCP 検証者であると考えられる. PCP 検証者の構成は非常に難しい. 例えば例 1.2.4 のグラフ彩色問題に対する次の検証者を考えてみよう: 入力としてグラフ $G = (V, E)$ と証拠として関数 $\pi\colon V \to [k]$ を受け取り, この関数 π が実際に G の k-彩色であるかどうかを判定する. 検証者は G の辺 $\{u, v\} \in E$ をランダムに一つ選び, $\pi(u) \neq \pi(v)$ であるかどうかによって判定する. この検証者は π のうち高々 $q(n) = O(1)$ 個の文字を読み, そのランダムシード長は $r(n) = O(\log n)$ で抑えられる. しかし, この検証者は健全性の条件を満たさない. 実際, G が k-彩色可能でない場合, どのような関数 $\pi\colon V \to [k]$ を与えても, 少なくとも一つの辺 $\{u, v\} \in E$ が存在して $\pi(u) = \pi(v)$ となるが, 検証者がこのような辺を引き当てる確率は最悪の場合, $1/|E|$ となるからである.

PCP 定理とは, ある $r(n) = O(\log n)$, $q(n) = O(1)$ に対して $\mathsf{PCP}(r, q) = \mathsf{NP}$ が成り立つことを主張する定理である. 例えばグラフ彩色問題は NP に属するため, 実は全段落の $r(n), q(n)$ を達成する PCP 検証者が存在するのである!

定理 1.2.7 （PCP 定理）

ある $r(n) = O(\log n)$, $q(n) = O(1)$ に対して $\mathsf{PCP}(r, q) = \mathsf{NP}$ が成り立つ.

注釈 1.2.8 （片側の包含関係）

PCP 定理において, $\mathsf{PCP}(r, q) \subseteq \mathsf{NP}$ は容易に示すことができる. 実際, 注釈 1.2.6 により, 証拠 π の長さは $q(n)2^{r(n)} = n^{O(1)}$ で抑えられる. また, $r(n) = O(\log n)$ より, 検証者は $2^{r(n)} = n^{O(1)}$ 個のランダムシードを列挙し, それら全てに対して PCP 検証者を適用し, その出力値の多数決をとることで, 入力 x に対して $x \in L$ かどうかを多項式時間で判定できる. PCP 定理の証明の本質的な難しさは逆側の包含関係 $\mathsf{NP} \subseteq \mathsf{PCP}(r, q)$ の証明にある.

1.2.3 3 彩色問題の NP 完全性

クラス NP に属する問題のうち最も難しいという性質を **NP 完全性**という. 本来は NP 完全性を定義するには帰着の概念を導入しなければならないが, この講義では 3 彩色問題が NP 完全であるという事実のみを用いるため, この事実について証明なしで述べる.

定理 1.2.9 （3 彩色問題の NP 完全性）

3 彩色問題 3COL は NP 完全である．すなわち，任意の $L \in \mathsf{NP}$ に対して，多項式時間アルゴリズム f が存在して，任意の $x \in \{0,1\}^*$ に対して $x \in L$ と $f(x) \in \mathrm{3COL}$ は同値である．

つまり，3COL を多項式時間アルゴリズムで解くアルゴリズムが存在するならば，NP に属する任意の判定問題 L を多項式時間で解くことができる．実際，定理 1.2.9 のアルゴリズム f を用いて L の入力を 3COL の入力に変換した後に，3COL を解くアルゴリズムを適用すればよい．この意味で，3COL は NP に属する問題の中で最も難しい問題 (の一つ) であるといえる．

注釈 1.2.10 （NP 完全性の定義）

この注釈は NP 完全性の定義を与えるものであり，この講義では必要ない．二つの判定問題 L, L' に対し，ある多項式時間アルゴリズム $f\colon \{0,1\}^* \to \{0,1\}^*$ が存在して，

$$x \in L \iff f(x) \in L'$$

を満たすとき，L は L' に**カープ帰着可能**といい，そのようなアルゴリズム f を**カープ帰着**という．このとき，L' を解く多項式時間アルゴリズムが存在すればそれを用いて L を解く多項式時間アルゴリズムを構成することができる．この意味で「L' は L 以上に難しい」とみなし，$L \leq_p L'$ と表す．
クラス NP に属するある判定問題 L は，任意の $L' \in \mathsf{NP}$ に対して $L' \leq_p L$ が成り立つとき，NP **完全**であるという．

3 彩色問題の NP 完全性より，PCP 定理 (定理 1.2.7) を証明するには 3 彩色問題に対して PCP 検証者を構成すればよいことがわかる．

定理 1.2.11 （3 彩色問題の PCP 検証者）

ある $r(n) = O(\log n), q(n) = O(1)$ に対して，3 彩色問題 3COL は $\mathsf{PCP}(r, q)$ に属する．

演習問題 2

定理 1.2.11 と定理 1.2.9 を仮定して，PCP 定理 (定理 1.2.7) を証明せよ．

1.3 PCP 定理の歴史

PCP 定理は計算複雑性理論において重要な成果の一つである．1997 年，Raz と Safra は低次多項式の制限を使用する新しい低次テストを導入し，NP の低エラー PCP 特性を証明した．この研究により，任意の定数 $c > 0$ に対して，任意の NP 言語のメンバーシップは $O(1)$ 回のアクセスで検証でき，各アクセスは対数的なビット数を読み取り，エラー確率は $2^{-(\log n)^c}$ となることが示された．

1998 年には, Arora と Safra が NP の新しい特性付けを提供し, NP はメンバーシップの証明が対数的な数のランダムビットを使用し, 証明から部分対数的な数のビットを読み取ることによって, 確率的に多項式時間で検証できる言語のクラスを正確に含むことを示した. また, Arora らは, NP のすべての言語が対数的な数のランダムビットを使用し, 証明内の定数数のビットを調べることによってメンバーシップ証明をチェックする確率的な検証者を持つことを証明した. この結果は NP $\not\subseteq$ P でない限り, MAX SNP 困難な問題は多項式時間近似スキームを持たないことを示している.

1.3.1 PCP 定理の最近の研究動向

PCP 定理の証明以降, 研究者たちは PCP システムの効率性と応用可能性を向上させるための様々な研究を行ってきた. 特に注目すべき最近の研究動向は以下の通りである:

1. **効率的な PCP の構築**:
   - Ben-Sasson and Sudan [BS06] は, 準線形長 (almost-linear length) の PCP を構築し, 証明長を大幅に短縮した.
   - Ben-Sasson and Sudan [BS06] は, 多項式対数 (polylog) の問い合わせ複雑性を持つ短い PCP を提案した.
2. **低エラー PCP の研究**:
   - Dinur and Harsha [DH13] は, 復号可能な PCP(Decodable PCPs) の概念を導入し, 低エラー 2 クエリ PCP の合成方法を開発した.
   - Guruswami, Harsha, Sudan, and Zhou [GHSZ20] は, 定数の健全性を持ちながら低エラー確率を実現する PCP において, アルファベットサイズを削減する新しい技術を提案した.
3. **ゼロ知識 PCP の研究**:
   - Gur, Liu, and Rothblum [GLR24] は, #P 内のすべての言語に対する完全ゼロ知識 PCP を初めて構築した.
   - さらに, 任意の多項式時間敵対者に対して完全ゼロ知識である NEXP 用の指数サイズ定数クエリ PCP も構築した.
4. **新しい応用分野の開拓**:
   - Abboud, Rubinstein, and Williams [ARW17] は, 分散 PCP の概念を導入し, 強指数時間仮説 (SETH) から P 内の近似問題への初めての PCP 的還元を提供した.
   - Chen and Williams [CW19] は, fine-grained complexity と対話型証明システムの理論を融合させ, 計算問題の細粒度な困難性に関する新しい知見を提供した.

これらの研究は, PCP の理論的基盤を強化するとともに, 暗号プロトコルや対話型証明システムなどの応用分野にも新たな可能性を開いている. 特に, ゼロ知識 PCP の研究は, プライバシーを保持しながら効率的な証明検証を可能にする新しい方向性を示している.

Chapter 2

制約充足問題

制約充足問題 (Constraint Satisfaction Problem, CSP) は, 計算量理論において重要な問題の一つであり, PCP 定理の証明においても中心的な役割を果たす.

2.1 制約充足問題の定義

制約充足問題とは端的に言えば連立方程式の解の存在性判定を問う判定問題である.

定義 2.1.1 （制約充足問題）

制約充足問題 (CSP) とは次の要素からなる組 $\varphi = (X, \Sigma, \mathcal{I}, \mathcal{C})$ を入力とする判定問題である:

- **アルファベット**と呼ばれる有限集合 Σ.
- 変数集合 $X = \{x_1, \ldots, x_n\}$.
- 部分集合族 $\mathcal{I} = \{I_1, \ldots, I_m\}$. ただし各 I_i は $I_i \subseteq [n]$ である.
- 関数族 $\mathcal{C} = \{c_1, \ldots, c_m\}$. ただし各 c_i は $c_i : \Sigma^{I_i} \to \{0, 1\}$ である.

入力 $(X, \Sigma, \mathcal{I}, \mathcal{C})$ は, ある変数への割り当て $(a_1, \ldots, a_n) \in \Sigma^n$ が存在して, 任意の $i \in [m]$ について $c_i(a_{I_i}) = 1$ であるとき, かつその時に限り Yes インスタンスである. ここで, $a_{I_i} = (a_{i_1}, \ldots, a_{i_{|I_i|}})$ は a_i の部分列である. 特に, 全ての $i \in [m]$ について $|I_i| \leq q$ であるとき, この CSP は **q-CSP** という.
また, 固定した割り当て $\vec{a} = (a_1, \ldots, a_n)$ に対する φ の**充足度**を

$$\mathrm{val}(\vec{a}) = \Pr_{i \sim [m]}[c_i(\vec{a}_{I_i}) = 1]$$

とし, 全ての割り当てに関して充足度の最大値

$$\mathrm{val}(\varphi) = \max_{\vec{a} \in \Sigma^n} \mathrm{val}(\vec{a})$$

を φ の充足度という.

例 2.1.2 （グラフ彩色問題）

グラフ彩色問題 (例 1.2.4) は 2-CSP である. 実際, グラフ $G = (V, E)$ に対して

- 変数集合を $X = V$ とする.
- アルファベットを $\Sigma = [k]$ とする.
- 制約集合を $\mathcal{I} = \{E\}$ とする.
- 関数族を $\mathcal{C} = \{c_e\}_{e \in E}$ とし, 各 $e = \{u, v\} \in E$ に対して

$$c_e(a_u, a_v) = \begin{cases} 1 & \text{if } a_u \neq a_v, \\ 0 & \text{if } a_u = a_v \end{cases}$$

  と定義する.

このとき, グラフ G が k-彩色可能であることと, この CSP が Yes インスタンスであることは同値である.

2.1.1 PCP との関係

ランダムシード長 $r = r(n)$ かつクエリ数 $q = O(1)$ の PCP 検証者は $\Sigma = \{0, 1\}$ の場合の q-CSP によって表現でき, 逆に q-CSP は PCP 検証者として表現できる. 実際, q クエリの PCP 検証者 $V^\pi(x)$ を考えよう. 証明の長さを $|\pi| = \ell$ とする. 入力 x とランダムシード s を固定したときの $V^\pi(x; s)$ が証明中の読み込む文字のインデックスの集合を $I_s \subseteq [\ell]$ とする (ここで $|I_s| \leq q$). このとき, $V^\pi(x; s)$ は $\{0, 1\}^{I_s}$ を $\{0, 1\}$ に写す関数を定める. この関数を制約 c_s とみなすことで, 検証者 $V^\pi(x)$ は q-CSP のインスタンスとして表現できる. このとき, q-CSP のインスタンスの変数集合は証明 π に対応する. もし $x \in L$ であるならば, 確率 1 で $V^\pi(x) = 1$ となるような π が存在する. つまり, 全てのランダムシード s に対して $V^\pi(x; s) = 1$ となるような π が存在するため, 先ほど構成した q-CSP のインスタンスは Yes インスタンスである. 逆に $x \notin L$ であるならば, 全ての π に対して確率 $1/3$ 以上で $V^\pi(x) = 0$ となる. これは, q-CSP インスタンスに対して, 全ての割り当てを考えても, 全体の制約のうち少なくとも $1/3$ の割合は充足されないことを意味する.

PCP 検証者	q-CSP
PCP π	割り当て
ランダムネスを固定した時の判定	CSP の制約
PCPπ を受理する確率	割り当ての充足度 $\mathrm{val}(\pi)$

Table 2.1: PCP と CSP の対応関係

この対応関係に基づいて, PCP 定理を CSP を用いた言葉で表すことができる.

定理 2.1.3 （PCP 定理の CSP 版）

ある関数 $m = n^{O(1)}$, $q = O(1)$, $\ell = n^{O(1)}$, 定数 $c \in \mathbb{N}$, $\varepsilon > 0$, および次の性質を満たす多項式時間決定的アルゴリズム A が存在する: 3 彩色問題のインスタンス $G = (V, E)$ を入力として受け取り, G の頂点数を n としたとき, A は高々 $\ell(n)$ 個の変数と $m(n)$ 個の制約および要素数 c のアルファベットからなる q-CSP のインスタンス φ を出力する. さらにこのインスタンス φ は

- 入力 G が 3COL の Yes インスタンスであるとき, $\mathrm{val}(\varphi) = 1$ となる (すなわち φ は Yes インスタンス).
- 入力 G が 3COL の No インスタンスであるとき, $\mathrm{val}(\varphi) < 1 - \varepsilon$ となる.

補題 2.1.4

定理 2.1.3 と定理 1.2.11 は同値である.

証明. それぞれの方向を別々に証明する.

定理 2.1.3⇒ 定理 1.2.11 の証明. ある $r = O(\log n)$, $q' = O(1)$ に対して, 3COL に対するシード長 r, クエリ回数 q' の PCP 検証者 V^π を構成する. 入力としてグラフ $G = (V, E)$ を受け取り, 定理 2.1.3 のアルゴリズム A を用いて q-CSP のインスタンス φ を出力する. また, PCP π はこの q-CSP のインスタンス φ の割り当てとして解釈し, PCP 検証者 V^π は以下の操作を十分大きな定数回繰り返す: 一様ランダムに制約 c_i を選択し, その制約に含まれる変数に対する割り当て π の値を読み込み, この制約が充足**されない**ならば 0 を出力し終了する. 何度も繰り返した末に終了しなかったのであれば, 1 を出力して終了する. この検証者は繰り返しの回数が $O(1)$ であり, それぞれの繰り返しにおいては高々 q 個の変数の値を読み込むため, クエリ回数は $q' = O(q) = O(1)$ となる. なお, ここでアルファベットサイズ c が定数であることに留意する (実際には PCP π は二進文字列なので, 変数割り当てを読み込む際には $\log_2 c$ 文字を読み込んでいる). また, ランダムシードはランダムな制約を選ぶために使われてるため, その長さは $O(\log m) = O(\log n)$ となる.

もしグラフ G が Yes インスタンスならば, φ も Yes インスタンスであるため, π をその充足割り当てとすれば, 全ての制約 c_i が充足されるため, (制約の選び方のランダムネスに関して) 確率 1 で $V^\pi(G) = 1$ となる. もしグラフ G が No インスタンスならば, $\mathrm{val}(\varphi) \leq 1 - \varepsilon$ である. 従って, 任意の割り当て π に対して, 一様ランダムな制約 c_i が充足される確率は高々 $1 - \varepsilon$ である. よって, この操作を $\lceil 10/\varepsilon \rceil = O(1)$ 回繰り返すと, 少なくとも確率 $1/3$ で充足されない制約が一度以上選ばれ, 検証者は 0 を出力する.

定理 1.2.11⇒ 定理 2.1.3 の証明. 仮定より, 3COL に対する, ランダムシード長 $r = O(\log n)$, クエリ回数 $q = O(1)$ の PCP 検証者 V^π が存在する. アルゴリズム A は, 入力 G に対して, 全てのランダムシード $s \in \{0,1\}^r$ を列挙して, それぞれの $V^\pi(G; s)$ を関数 c_s とみなして, これらを制約とする CSP を出力する. 各 $V^\pi(G; s)$ は, π を変数とみなしたとき, 高々 q 個の変数の値を読み込むため, $(c_s)_{s \in \{0,1\}^r}$ は $2^r = n^{O(1)}$ 個の制約からなる q-CSP となる. なお, V^π は多項式時間アルゴリズムなので, A も多項式時間アルゴリズムである. □

従って, 以降は定理 2.1.3 の証明に注力する.

2.2 PCP 定理の証明の概要

PCP 定理の証明は, 制約システムの不満足値と呼ばれる量に着目する. 不満足値とは, 変数への全ての可能な割り当てについて, 満たされない制約の最小の割合を表す. この証明では, 以下の 2 つの主要なステップを繰り返し適用する:

1. **増幅変換**: 制約システムの不満足値を 2 倍に増幅する. この変換は, システムのサイズを線形に増加させるのみである. この増幅ステップは, 制約システムの基礎となるグラフ構造がエクスパンダーであることを利用する.

2. **PCP 合成**: 増幅ステップによって増加したアルファベットサイズを修正する. これは標準的な PCP 合成ステップを用いる.

これらの 2 つのステップを反復的に適用することで, PCP 定理の証明が得られる. このアプローチの特徴は, 制約システムの不満足値という量に着目することで, より直接的に PCP 定理を証明できる点にある.

2.3 制約グラフ

制約グラフは, 変数の集合とその変数間の制約を表現するグラフ構造であり, PCP 定理の証明において中心的な役割を果たす.

定義 2.3.1 （制約グラフ）

$G = \langle (V, E), \Sigma, C \rangle$ は以下の条件を満たすとき, **制約グラフ**と呼ばれる:

1. (V, E) は無向グラフであり, G の基礎となるグラフである.
2. 集合 V は, アルファベット Σ 上の値を取る変数の集合としても見なされる.
3. 各辺 $e \in E$ には制約 $c(e) \subseteq \Sigma \times \Sigma$ が付随し, $C = \{c(e)\}_{e \in E}$ である.

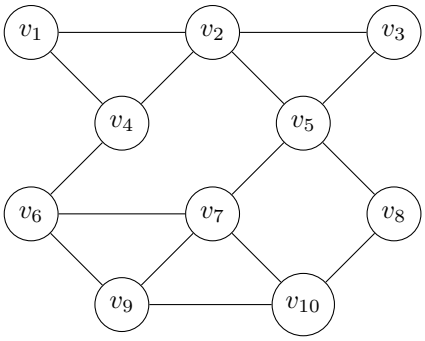
割り当て $\sigma : V \to \Sigma$ に対して, $\mathrm{UNSAT}_\sigma(G)$ は割り当て σ によって満たされない辺の割合を表す:

$$\mathrm{UNSAT}_\sigma(G) = \Pr_{(u,v) \in E}[(\sigma(u), \sigma(v)) \notin c(e)]$$

そして, $\mathrm{UNSAT}(G) = \min_\sigma \mathrm{UNSAT}_\sigma(G)$ を G の**不満足値**と呼ぶ.

2.3.1 前処理補題

PCP 定理の証明の最初のステップは, 任意の CSP インスタンスを制約グラフに変換する前処理補題である. この変換は, 元の CSP の充足可能性と不満足値の基本的な性質を保持しつつ, 制約グラフの形式に変換する.



$\Sigma = \{1, 2, 3\}$
$c(e) = \{(i, j) \in \Sigma^2 : i \neq j\}$
$\mathrm{UNSAT}(G) = \frac{1}{15}$

図 2.1: 3 彩色問題の制約グラフの例. 各頂点は 3 色のいずれかで彩色される必要があり, 隣接する頂点は異なる色でなければならない. この例では最適な彩色でも少なくとも 1 つの辺の制約を違反する必要があり, その割合は $\frac{1}{15}$ となる.

補題 2.3.2 （前処理補題）

任意の q-CSP インスタンス φ に対して, 以下の性質を満たす制約グラフ G を多項式時間で構築できる:

- φ が Yes インスタンスであることと, $\mathrm{UNSAT}(G) = 0$ であることは同値である.
- φ が No インスタンスであるとき, $\mathrm{UNSAT}(G) \geq \frac{1-\mathrm{val}(\varphi)}{q}$ である.
- グラフ G の頂点数は $|V| = O(n)$ である.
- グラフ G の辺数は $|E| = O(m)$ である.
- アルファベット Σ は元の CSP と同じである.

この補題の証明は, CSP インスタンスの制約を辺の制約に変換する単純な構築法に基づく. 具体的には, 各制約 c_i に対して, その制約に含まれる変数間の辺を追加し, その辺に制約 c_i を付随させる. この変換により, CSP の充足可能性と不満足値の基本的な性質が保持される.

前処理補題は, 任意の CSP インスタンスを制約グラフの形式に変換することで, 以降のギャップ増幅ステップのための準備を行う. この変換は多項式時間で実行可能であり, 元の CSP の重要な性質を保持するため, PCP 定理の証明の基礎となる.

2.4 エクスパンダーグラフ

2.4.1 エクスパンダーグラフの定義

エクスパンダーグラフは, グラフの接続性を定量的に表現する重要な概念である. 特に, 任意の部分集合に対して, その境界のサイズが部分集合のサイズに応じて十分大きいという性質を持つ.

定義 2.4.1 （エクスパンダーグラフ）

d-正則グラフ $G = (V, E)$ が (n, d, λ)-**エクスパンダー**であるとは, 以下の条件を満たすことである:

- $|V| = n$ である.
- 任意の頂点の次数が d である.
- 任意の $S \subseteq V$ に対して, $|S| \leq n/2$ のとき

$$\left|E(S, \overline{S})\right| \geq \lambda \cdot d \cdot |S|$$

が成り立つ. ここで, $E(S, \overline{S})$ は S と $\overline{S}$ を結ぶ辺の集合である.

エクスパンダーグラフの重要な性質として, ランダムウォークの収束が速いことが知られている. これは, エクスパンダーグラフのスペクトルギャップが大きいことと密接に関連している.

2.4.2 エクスパンダー混交補題

エクスパンダー混交補題は, エクスパンダーグラフにおけるランダムウォークの収束性を定量的に表現する重要な結果である.

補題 2.4.2 （エクスパンダー混交補題）

(n, d, λ)-エクスパンダー $G = (V, E)$ を考える. 任意の $S, T \subseteq V$ に対して

$$|E(S, T)| - \frac{d}{n}|S||T| \leq \lambda\sqrt{|S||T|}$$

が成り立つ.

証明. 隣接行列 A の固有値を $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ とする. このとき, $\lambda_1 = d$ であり, $\lambda_2 \leq \lambda$ であることが知られている.

任意の $S, T \subseteq V$ に対して, 特性ベクトル χ_S, χ_T を考える. このとき

$$|E(S, T)| = \chi_S^\top A \chi_T$$

が成り立つ. また, χ_S と χ_T を A の固有ベクトルで展開すると

$$\chi_S = \sum_{i=1}^{n} \alpha_i v_i, \quad \chi_T = \sum_{i=1}^{n} \beta_i v_i$$

となる. ここで, v_i は A の固有ベクトルである.

このとき

$$\begin{aligned}
|E(S,T)| - \frac{d}{n}|S||T| &= \chi_S^\top A \chi_T - \frac{d}{n}|S||T| \\
&= \sum_{i=1}^{n} \lambda_i \alpha_i \beta_i - \frac{d}{n}|S||T| \\
&= \sum_{i=2}^{n} \lambda_i \alpha_i \beta_i \\
&\leq \lambda \sum_{i=2}^{n} |\alpha_i \beta_i| \\
&\leq \lambda \sqrt{\sum_{i=2}^{n} \alpha_i^2} \sqrt{\sum_{i=2}^{n} \beta_i^2} \\
&\leq \lambda \sqrt{|S||T|}
\end{aligned}$$

が成り立つ. □

エクスパンダー混交補題は, エクスパンダーグラフにおける辺の分布が一様であることを示している. これは, PCP 定理の証明において, 制約システムの不満足値を増幅する際に重要な役割を果たす.

2.5 制約グラフのエクスパンダー化

制約グラフのエクスパンダー化は, PCP 定理の証明における重要なステップである. この変換により, 制約グラフの不満足値を増幅することが可能となる.

補題 2.5.1 （制約グラフのエクスパンダー化）

制約グラフ $G = \langle (V, E), \Sigma, C \rangle$ に対して, 以下の性質を満たす制約グラフ $G' = \langle (V', E'), \Sigma, C' \rangle$ を多項式時間で構築できる:

- 基礎グラフ (V', E') は (n', d', λ')-エクスパンダーである. ここで, $n' = O(n)$, $d' = O(1)$, $\lambda' = O(1/\sqrt{d'})$ である.
- $\mathrm{UNSAT}(G') \geq \Omega(\mathrm{UNSAT}(G))$ である.
- アルファベット Σ は元のグラフと同じである.

証明. 制約グラフ G をエクスパンダーに変換する方法を説明する. 主なアイデアは, グラフの冪乗操作を用いて接続性を向上させることである.

グラフの冪乗操作: 制約グラフ G の t 乗 G^t を以下のように定義する:

- 頂点集合は元のグラフと同じ V である.
- 辺 (u, v) は, G 上で u から v への長さ t のパスが存在する場合に存在する.

- 各辺 e' の制約 $c'(e')$ は, 対応するパス上の制約の合成である. すなわち, パス $e_1, \ldots, e_t$ に対して

  $$c'(e') = \{(a,b) \in \Sigma^2 : \exists a_1, \ldots, a_{t-1} \in \Sigma, (a, a_1) \in c(e_1), (a_1, a_2) \in c(e_2), \ldots, (a_{t-1}, b) \in c(e_t)\}$$

  と定義する.

この冪乗操作により, グラフの接続性が向上し, エクスパンダー性が得られる. 具体的には, 以下の性質が成り立つ:

補題 2.5.2

制約グラフ G の t 乗 G^t は, 元のグラフ G の接続性を t 倍に増幅する. すなわち, 任意の $S \subseteq V$ に対して

$$\left|E_{G^t}(S, \overline{S})\right| \geq t \cdot \left|E_G(S, \overline{S})\right|$$

が成り立つ.

Lemma の証明. 任意の $S \subseteq V$ を考える. G 上の S と $\overline{S}$ を結ぶ辺の数を k とする. このとき, G^t 上では, これらの辺を通る長さ t のパスが少なくとも k 個存在する. 各パスは G^t 上の辺に対応するため, $E_{G^t}(S, \overline{S})$ のサイズは少なくとも k である. □

次に, 不満足値の関係を証明する:

補題 2.5.3

$\mathrm{UNSAT}(G^t) \geq \Omega(t \cdot \mathrm{UNSAT}(G))$ が成り立つ.

Lemma の証明. 任意の割り当て $\sigma : V \to \Sigma$ を考える. G 上の不満足な辺の割合を ε とする. このとき, G^t 上の任意の辺 e' は, t 個の G 上の辺の制約の合成である. これらの制約のうち少なくとも εt 個が不満足であるため, $\mathrm{UNSAT}_\sigma(G^t) \geq \Omega(\varepsilon t)$ が成り立つ. □

最後に, 適切な t を選択することで, 所望のエクスパンダー性が得られることを示す. $t = O(\log n)$ とすると, グラフ G^t は (n', d', λ')-エクスパンダーとなる. ここで, $n' = n$, $d' = d^t = O(1)$, $\lambda' = O(1/\sqrt{d'})$ である. □

この補題により, 任意の制約グラフをエクスパンダーに変換しつつ, 不満足値を保持することができる. これは, PCP 定理の証明におけるギャップ増幅ステップの基礎となる.

Chapter 3

ギャップ増幅補題

この章ではPCP定理の証明において重要な役割を果たすギャップ増幅補題について解説する.

3.1 主張と直感

ギャップ増幅補題は, 制約グラフの不満足値を増幅するための重要なツールである. この補題は, エクスパンダーグラフの性質を利用して, 制約グラフの不満足値を2倍に増幅する.

補題 3.1.1 （ギャップ増幅補題）

定数$\alpha > 0$が存在し, 以下の性質を満たす多項式時間アルゴリズムが存在する: 入力として制約グラフ$G = \langle (V, E), \Sigma, C \rangle$を受け取り, 新しい制約グラフ$G' = \langle (V', E'), \Sigma', C' \rangle$を出力する.

- $\mathrm{UNSAT}(G) = 0$ならば$\mathrm{UNSAT}(G') = 0$である.
- $\mathrm{UNSAT}(G) > 0$ならば$\mathrm{UNSAT}(G') \geq \min(2\mathrm{UNSAT}(G), \alpha)$である.
- $|V'| = O(|V|)$である.
- $|E'| = O(|E|)$である.
- $|\Sigma'| = |\Sigma|^{O(1)}$である.

この補題の直感的な説明は以下の通りである:

- 制約グラフGの各頂点vを, その頂点の近傍の情報を持つ新しい頂点の集合に置き換える.
- 新しい頂点間の辺は, 元のグラフの辺の情報を保持するように設定する.
- エクスパンダーグラフの性質により, 元のグラフで満たされない制約は, 新しいグラフでより多くの制約違反を引き起こす.

3.2 証明

3.2.1 構成

ギャップ増幅補題の証明は, 以下の 3 つのステップから構成される:

1. **グラフの冪乗**: 制約グラフ G の冪乗 G^t を構築する. これは, 各頂点をその t-近傍の情報を持つ頂点に置き換える操作である.
2. **エクスパンダー変換**: 冪乗グラフ G^t を, エクスパンダーグラフの性質を持つ新しいグラフ G' に変換する.
3. **制約の変換**: 新しいグラフ G' の各辺に, 元の制約を適切に変換した制約を付随させる.

3.2.2 正当性の証明

ギャップ増幅補題の正当性は, 以下の 2 つの重要な性質に基づいて証明される:

1. **完全性**: $\mathrm{UNSAT}(G) = 0$ の場合, 新しいグラフ G' も $\mathrm{UNSAT}(G') = 0$ を満たす. これは, 元のグラフの充足割り当てを新しいグラフに自然に拡張できることから従う.
2. **健全性**: $\mathrm{UNSAT}(G) > 0$ の場合, エクスパンダーグラフの性質により, 新しいグラフ G' の不満足値は元のグラフの 2 倍以上になる. これは, エクスパンダーグラフの混交補題を用いて証明される.

これらの性質により, ギャップ増幅補題が成り立つことが示される. この補題は, PCP 定理の証明において, 小さな不満足値を大きな値に増幅するための重要なツールとなる.

Chapter 4

アルファベット削減

4.1 PCP 定理の証明

PCP 定理の証明の最後のステップは, アルファベットサイズを定数に削減することである. これは, ギャップ増幅ステップによって増加したアルファベットサイズを, 標準的な PCP 合成ステップを用いて修正する.

補題 4.1.1 （アルファベット削減補題）

定数 $\varepsilon > 0$ が存在し, 以下の性質を満たす多項式時間アルゴリズムが存在する: 入力として制約グラフ $G = \langle (V, E), \Sigma, C \rangle$ を受け取り, 新しい制約グラフ $G' = \langle (V', E'), \Sigma', C' \rangle$ を出力する.

- $\mathrm{UNSAT}(G) = 0$ ならば $\mathrm{UNSAT}(G') = 0$ である.
- $\mathrm{UNSAT}(G) > 0$ ならば $\mathrm{UNSAT}(G') \geq \min(\mathrm{UNSAT}(G), \varepsilon)$ である.
- $|V'| = O(|V|)$ である.
- $|E'| = O(|E|)$ である.
- $|\Sigma'| = O(1)$ である.

この補題の証明は, 標準的な PCP 合成ステップを用いる. 具体的には:

1. **制約の分解**: 各制約 $c(e)$ を, より小さなアルファベットサイズを持つ制約の集合に分解する.
2. **制約の合成**: 分解された制約を, 新しい制約グラフ G' の制約として合成する.
3. **正当性の証明**: 元の制約グラフと新しい制約グラフの不満足値の関係を証明する.

このアルファベット削減補題は, PCP 定理の証明の最後のピースとなる. これにより, 制約グラフのアルファベットサイズを定数に削減しつつ, 不満足値の基本的な性質を保持することができる.

4.2 まとめ

PCP 定理の証明は, 以下の 3 つの主要なステップから構成される:

1. **前処理補題**: 任意の CSP インスタンスを制約グラフに変換する.
2. **ギャップ増幅補題**: 制約グラフの不満足値を 2 倍に増幅する.
3. **アルファベット削減補題**: アルファベットサイズを定数に削減する.

これらの補題を組み合わせることで, PCP 定理が証明される. この証明は, 制約システムの不満足値という量に着目することで, より直接的に PCP 定理を証明できる点に特徴がある.

Bibliography

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. "PRIMES is in P". en. In: **Annals of mathematics** 160 (2 Sept. 1, 2004), pp. 781–793. DOI: 10.4007/annals.2004.160.781 (cit. on p. 10).

[ALMSS98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. "Proof verification and the hardness of approximation problems". In: **Journal of the ACM** 45 (3 May 1, 1998), pp. 501–555. DOI: 10.1145/278298.278306 (cit. on p. 7).

[ARW17] A. Abboud, A. Rubinstein, and R. Williams. "Fine-grained reductions from approximate counting to decision". In: **Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing** (2017), pp. 281–290. DOI: 10.1145/3055399.3055400 (cit. on p. 14).

[AS98] S. Arora and S. Safra. "Probabilistic checking of proofs: a new characterization of NP". In: **Journal of the ACM** 45 (1 Jan. 1, 1998), pp. 70–122. DOI: 10.1145/273865.273901 (cit. on p. 7).

[BS06] E. Ben-Sasson and M. Sudan. "Short PCPs with polylog query complexity". In: **Proceedings of the thirty-eighth annual ACM symposium on Theory of computing** (2006), pp. 266–275. DOI: 10.1145/1132516.1132556 (cit. on p. 14).

[CW19] L. Chen and R. Williams. "Fine-grained complexity meets IP=PSPACE". In: **Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms** (2019), pp. 1–20. DOI: 10.1137/1.9781611975482.1 (cit. on p. 14).

[DH13] I. Dinur and P. Harsha. "Composition of low-error 2-query PCPs using decodable PCPs". In: **SIAM Journal on Computing** 42.6 (2013), pp. 2452–2486. DOI: 10.1137/120875636 (cit. on p. 14).

[Din07] I. Dinur. "The PCP theorem by gap amplification". In: **Journal of the ACM** 54 (3 June 1, 2007), 12–es. DOI: 10.1145/1236457.1236459 (cit. on p. 7).

[GHSZ20] V. Guruswami, P. Harsha, M. Sudan, and S. Zhou. "Constant rate PCPs for circuit-SAT with sublinear query complexity". In: **Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing** (2020), pp. 9–20. DOI: 10.1145/3357713.3384244 (cit. on p. 14).

[GLR24] T. Gur, T. Liu, and R. D. Rothblum. "Perfect Zero Knowledge: New Upper Bounds and Relativized Separations". In: **arXiv preprint arXiv:2401.00001** (2024) (cit. on p. 14).