プログラムコード生成 AI と著作権を巡る問題

―GitHub Copilot に係る紛争からの示唆―

青山学院大学法学部法学科　教授　**平嶋　竜太**

要　約

生成 AI と知的財産法を巡っては、目下、多種多様な生成物の法的取扱い及び権利侵害といった問題を中心として、知的財産法各法の領域にわたり様々な議論が生じている状況にある。もっとも、コンピュータ・プログラム言語によるプログラムコードの生成を行う AI（コード生成 AI）と著作権法を巡る法的議論については、コンテンツ（画像、音楽等）を生成する AI と著作権法に関する議論と比べると、現状では未だ十分な検討・議論がなされていない状況にあるといえるものの、アメリカでは、コード生成 AI と著作権に関する民事訴訟が連邦地裁に係属中であり、法的リスクが表面化しつつある。他方で、コード生成 AI の発展・普及が、今後のソフトウエア産業の在り方に対してもたらすインパクトは極めて大きいものであると考えられる。このことは、コード生成 AI と著作権法に係る問題を契機として、プログラムを著作物として法的保護を与えることを所与としてきた現行のプログラム保護法制自体の在り方を再考することに繋がる可能性もあるものと考えられる。

以上の問題意識から、本稿では、アメリカにおいて訴訟審理が目下進行中である、GitHub Copilot を巡る事案の検討を糸口として、コンテンツ生成 AI との比較、日本法の下で想定される著作権法上の課題について検討して、コード生成 AI と著作権法を巡る法的議論に係る考察を行う。これらを踏まえて、現行の知的財産法制の下でのコンピュータ・プログラムの保護の在り方自体に対する将来的な方向性についても若干の展望を加えるものである。

目　次

1．はじめに

生成 AI[(1)]と知的財産法を巡っては、目下、多種多様な生成物の法的取扱い及び権利侵害問題等を巡って、知的財産法各法の領域で様々な議論が生じている状況にある[(2)]。現状では、コンテンツ（画像、音楽等）を主に生成する AI と著作権法に関する議論に比べて、コンピュータ・プログラム言語で記述されたプログラムコードを生成をする AI（以下、コード生成 AI とする）と著作権法を巡る法的議論については、世界的にもそれほど耳目を集めてはいない状況にあるといえるものの、アメリカでは、コード生成 AI と著作権に

(1)　元々は、generative AI という語の訳語であって、本稿の基となる報告時では生成系 AI という語を用いていたが、至近の状況（本稿執筆時である 2024 年 4 － 6 月時点）では、日本語の文献等々では生成系 AI よりも生成 AI という訳語の方が多く用いられる状況にあることから、本稿でもこの語を用いることにする。

(2)　各法における課題について、先に筆者は著作権法・特許法を中心に全体的な課題の俯瞰を行った（平嶋竜太・Generative AI による生成物を巡る知的財産法の課題・別冊 L & T9 号 61 － 75 頁）。もっとも、同稿公表後、既に 1 年程経過しており、生成 AI 関連技術における進展の速さに伴い、同稿では十分に把握し切れていない事項も新たに顕在化しているものと考えられる。

関する法的紛争[3]がカリフォルニア州北部地区連邦地裁に現在係属中であり、法的リスクが表面化していることは明らかである。コンテンツ系の生成 AI と比べてコード生成 AI 固有の法的論点を見出すことができるのか、という点で理論的に注目されるところであって、産業における実質的なインパクトという側面からは、コード生成 AI がソフトウエア産業の在り方へ将来的にもたらす影響は極めて大きいものとも考えられる。さらには、この問題を契機に、プログラムを著作物として法的に保護することを所与とする現行のプログラム保護法制自体の在り方を再考することにも繋がる問題であるものと考えられる。

以上の観点から、本稿ではコード生成 AI と著作権法を巡る法的議論について、アメリカで訴訟が目下進行中である GitHub Copilot を巡る法的紛争について検討し、コンテンツ生成 AI との比較、日本法の下で想定される著作権法上の課題について検討･考察する。これらの検討を踏まえて､現行の知的財産法制の下でのコンピュータ・プログラムの保護の在り方自体に対する将来的な方向性について若干の展望を加えるものである。

２．コード生成 AI の構造と現状

２－１　機械学習・深層学習を基礎とした AI の基本的なプロセス

コード生成 AI の技術的な構造と現状を理解する前提として、まず、近時もっとも活発な発展がみられる機械学習・深層学習を基礎とした人工知能（AI）の基本的な仕組みとプロセスについて、簡単に確認をしておくこととする。

機械学習・深層学習を基礎とした AI について、そのメカニズムを極めて概括的に把握するとすれば、あらゆる事象を入力と出力の間にある関係性として捉えた上で、両者の間を多数のパラメータより構成された関数をモデル[4]として設定・構築し、任意の入力に対応した最適な出力を得られうるモデルの構築を指向して得られた関数を前提として、任意の入力に対して推論する過程についてニューラルネットワークを用いて構成することによって実現がなされているといえる。そして、その推論過程においては基本的には確率過程としての処理が関わってくることが一般的といえる。どのようなモデルとして構築するのか、得られたモデルについてニューラルネットワークを基礎としてどのようなデータの処理を実現するのかというアーキテクチャの手法について、近時目覚ましい勢いで新しいモデルやアーキテクチャが開発・公開されているが、いずれも上記のような構成の下で、使用するモデルと推論を行う手段についてより優れたものを目指して開発が進められているものと理解できるであろう。

上記の過程において、最適なモデルの構築を実現するため、既に事象として入力と出力について把握されているデータを用いて行う一連の構築過程のことを「学習」と捉えることができる。優れたモデル構築を行うための「学習」には膨大かつ良質なデータが必要不可欠である。このようなモデルについては、一般的には予測系（モデル）と生成系（モデル）に大別されることもあるが､処理されている内容自体としては､データによる学習を通じて構築がなされたモデル（学習済モデル）を基にして、入力に対応して、予測される確率分布の推論を基に何らかの出力データを得るという意味ではいずれも同じであって、入力に対していかなる性質の出力を求めるのかという点で、モデルの構造や実装のためのアーキテクチャが異なるという点で両者の違いが生じてくるものと考えられる。

近時は、とりわけ生成系の分野が生成 AI とも呼ばれ、その発展・利用が著しく、とりわけ自然言語処理（NLP）を基に多様な用途に供しうる汎化したモデルとして構築された大規模言語モデル（LLM）を活用した生成 AI が全盛な状況にあることは周知のとおりである[5]。また、他方で、画像や動画といった形式の情

(3)　DOE 1 et. al. v. GitHub, Inc. et al., 4:22-cv-06823,（N.D. Cal.）
(4)　関数モデルと呼称されることもあるが、関数によって構成されていることを所与の前提として、以下ではモデルと称することにする。
(5)　本稿では、プログラムコードの生成に係る課題を取り扱うことから、LLM を中心としたモデルを主として念頭に置く。

報についての生成に用いるモデルとしては、至近では拡散モデルというモデルの活用が大きな成果を得ており、この領域における進展も非常に注目される(6)。

生成 AI として現在広く用いられているものとしては、基本原理とするモデルとして LLM をベースとするもの、それ以外の拡散モデルや GAN(7) などの学習モデルを利用するものを中心として、訓練データの種類やチューニングにおける違い等に応じて各種の生成データに特化した様々な生成 AI が開発されている。具体的には、汎用生成 AI として、ChatGPT、Bard、コンテンツ系のうち、画像生成では Midjourney、Stable Diffusion、Dall-E、音楽生成では、Jukebox、MusicLM、動画生成では、Stable Video Diffusion、Video Poet といったものがあり、コンピュータ・プログラムを生成するコード生成 AI もある。

上記のような機械学習・深層学習を基礎とした近時の AI はニューラルネットワークを基礎として実現されているが、ニューラルネットワーク内における情報の処理の流れの構成の在り方であるアーキテクチャにおける進歩も極めて重要といえる。大規模言語モデルをはじめとする様々なモデルの進展とともに、2017 年に Google の技術陣により開発･公表されたトランスフォーマー（Transformer）は､従前のアーキテクチャの延長線上の技術とはいえるものの、極めて革新的な仕組みであって、これを用いることで生成されるデータの質の向上に極めて大きな影響を及ぼしたとされる。トランスフォーマー（Transformer）における推論過程(8)では汎用性が高くなっており、その状態を踏まえて特定のタスクに係る入力に対応して推論された出力が得られる。トランスフォーマー（Transformer）については、大規模言語モデル（LLM）におけるアーキテクチャとして用いられることでの成果が大きく注目を集めることとなったが、言語情報の処理のみならず、多種多様な種類の情報についても入力に適応した出力を生成することは可能であって、人間の認識する段階では異なる種類のものとして認識される情報（音声、画像、テキスト、等々）であっても、深層学習の情報処理レベルでは、全て同質の単位（token）にまで抽象化されて処理されることになる。とりわけ、近時では、画像生成において、出力の質が高いことから注目を浴びている拡散モデルにおけるアーキテクチャとしても活用されて(9)おり、今後はそのような利用も拡大するものと考えられる。

大規模言語モデル（LLM）の特徴としては、特定のタスクに特化した学習モデルに比べて、人間の用いる自然言語についてコンピュータを利用して処理する自然言語処理（NLP）を基礎とすることから、多様な用途に供しうること、加えて、モデル構築に際してインターネット上で公表されているものを中心として膨大なデータセットを用いて数十億レベル以上の膨大な数のパラメータで構成されており、パラメータ数をより増大させて大規模モデル化するほど、汎化能力や学習の効率化が著しく図られることが期待できるようになった(10)。もっとも、大規模言語モデル（LLM）の場合、アーキテクチャとして前述のトランスフォーマー（Transformer）を用いることも相まって高い質の生成データが得られるようになった反面で、個別の入力ごとにモデル自体に対して再度学習が加えられてモデル全体に大きく変容が加えられた上で出力の生成がなされるわけではなく、あくまでも事前学習がなされた状態であることを基にしているため、モデル全体がアップデートされないことにより生じる問題（hallcination 等）も大きな課題として認識されている。もっともこのような技術的課題に対しても、大規模言語モデルへの入力段階で、補足的な情報を併せて入力する手法(11)

(6) また、さらには、両者を組み合わせたタイプの手法（LLM-grounded diffusion）も出現している。
(7) 敵対的生成ネットワーク（Generative Adversarial Network, GAN）と呼ばれる。
(8) 事前学習された深層生成モデルの下で、過去の学習により蓄積された情報が学習モデル内に「記憶」された状態が作り出されていることが、その機能向上に大きく影響しているものと解されているようである。
(9) トランスフォーマー（Transformer）をアーキテクチャーとして用いる拡散モデルについては、Diffusdion Transformer（DiPs）と呼ばれており、画像生成モデルとして、高い生成品質と計算効率を特徴として注目がなされているようである。
(10) パラメータ数の増加に伴って、汎化性能や学習の効率化が飛躍的に向上するという、両者の間に、いわゆる「べき乗則」が成り立つことが明らかとなったことによる（岡野原大輔『ディープラーニングを支える技術 2 －ニューラルネットワーク最大の謎』233、250 頁（技術評論社、2022））。
(11) RAG（Retrieval-Augmented Generation）等が代表的なものである。

や部分的な形で学習状態を修正させるといった手法（追加学習）による解消といった対応策が続々と案出されている。

２－２　コード生成 AI の現状

プログラムコードも特定のプログラミング言語の文法の下で構成される以上、広い意味では特定の言語体系の下で形成されるテキストの一種と捉えることも可能であることから、ChatGPT のような汎用生成系 AI ツールを用いてもプログラムコードを生成することは可能であるとされているが、プログラムコード生成に特化した生成 AI のモデルが近時数多く開発・公開されている。また、最近は、汎用大規模言語モデルでありつつも、コード生成にも適応することを強くアピールしているものも現れている状況にある(12)。

コード生成 AI の機能としては、コード自動生成、コード補完生成、自動バグ発見･修正といったものがある。

コード自動生成とは、文字通りプログラムコードを自動的に生成する機能であるが、現状では人間の何らの入力等も要しないような完全な自律生成を行うというものではなく、仕様を事前に特定する必要がある。また、生成されたコードについても、バグやセキュリティの問題が含まれうることになるため、自動生成されたコードを直ちに実装して利用できるとは限らない。例えば、後に触れる OpenAI による Codex や本稿で以下扱う事案に係る GitHub の Copilot もこのような自動生成機能に留まるものである。GitHub の Copilot では、大規模言語モデル GPT-3 ベースに主としてプログラミング言語 Python のコードを用いて学習がなされたものであるが、Python 以外の言語のコードの生成も可能とされる。

コード補完生成とは、ユーザのコード記述をサポートする形で、次に予測されるコード記述を候補として提示・補完する機能である。前記のコードの自動生成までの自動化を行うものではないが、ある意味でコード生成の基本となる機能といえる。

自動バグ発見・修正とは、プログラムコード内のバグを自動的に検出・修正を行うことによって、プログラム開発者の負担軽減・支援を行う機能である。

コード生成系 AI ツールの具体例としては、例えば以下のようなものが挙げられる。GitHub Copilot は、GitHub(13)及び OpenAI(14)により開発されたものであって、基本的にはコード補完ツールをベースとするものである。Code Lllama は、Meta によって開発されている大規模言語モデルを基に GitHub のコードで学習を行ったものであってコード生成に特化している。Codey は、Google によって 大規模言語モデル PaLM 2 ベースに開発されたものである。その他、中国を中心に開発されて急速に展開しているとされる CodeGeeX、アマゾンのプラットフォーム AWS との接続性・統合性のあるコード生成を特徴とする Amazon CodeWhisperer、タスクに対する最適化アルゴリズムを生成するコード自動生成サービスとされる AI Programmer、コード自動補完ツールであるが、コード生成も可能とされる TabNine(15)、ChatGPT ベースでコードの自動生成・自動補完を行う Hugging Face、プログラミング支援ツールの一機能としてコード生成を補完する IntelliCode（Microsoft）、オープンソースコードのバク発見･修正提案を行う Deepcode といったものがみられる(16)。

画像・音楽等のコンテンツ生成 AI が入力としてのプロンプトに対応した画像情報や音声情報を生成して出

---

(12) Llama, Llama2 とは、Meta の開発した LLM であって、GitHub のレポジトリも用いて学習がなされている。これを基にコード生成にさらに特化した Code Llama も開発されている。その他、Google により Gemini が 2023 年 12 月に発表されており、これも基本的には汎用大規模言語モデルであるが、コード生成にも強力な性能を有しているものとされている。

(13) 現在は Microsoft 子会社である。

(14) 当初は非営利法人であったが、実質的に Microsoft との資本関係を有する法人となっている。

(15) Codota（イスラエル）によって開発されている。コード生成を自動化することは可能であるが、予め仕様を特定しておく必要があるとされている。

(16) さらに、ごく近時では、ソフトウエアの設計書を入力することによって、コードを自動生成するアプリである CodeAGI のように API を通じて生成 AI を活用するツールも現れてきているようである。

力データとしているのに対して、コード生成 AI では、プロンプトに対応したプログラムコードを生成して、出力データとしているものの、生成プロセスにおける基本原理は同様であって、コード生成 AI では、学習データとして既存のプログラムコードの膨大な集積（レポジトリ）及び各種プログラム言語（文法）を用いて訓練することでモデル構築がなされることで、プロンプトに対応した最適解とされるプログラムコードを推論の上で生成・出力するものである。その点では、近時急激な拡がりをみせる、ChatGPT、Bard 等に代表される大規模言語モデル（LLM）を基にした深層学習モデルと実質的には同様で、生成対象についてプログラムコード生成に専門分化したものであって、現時点では、完全に自動化されたコード生成を実現するものではない。

他方、コード生成 AI とは別の流れであるプログラミング自動化手法として、ノーコード（No Code）という流れについても注意し、両者を峻別しておくことを要するといえる。ノーコードについては、画面操作等によって、コードを生成する手法であって、プログラミング言語の知識を必ずしも要することなくソフトウエア開発を（ある程度）可能とするものである。

コード生成 AI の限界及びデメリットとしては、次のようなものが挙げられている。

まず、現時点における限界としては、完全に自動化されたコード生成を実現するものではなく、生成・出力されたプログラムコードが動作しない場合や誤ったものであることも珍しくないようである。このため、生成後も人間が確認して手を加えることは基本的に避けられず、生成されたコードについては、セキュリティ上の脆弱性が高い場合も生じうることから、生成されたコードを改めて人間によりチェックする必要性は依然として残っている状況にあると考えられる。プログラム開発者の視点からは、現状ではあくまで人間によるプログラミングのサポートや補完のためのツールという位置付けという受け止め方も強いものとみられる。

他方で、コード生成 AI のメリットとしては、ソフトウエア開発の生産性向上が大きく期待されており、パターン化された内容のコード作成作業等に生成 AI ツールを導入することでソフトウエア開発のスピードを大幅に削減できる可能性が期待できる。また、人間は単純なコード作業に関わることから解放され、より重要性の高い異なる領域の開発作業に注力することで、ソフトウエア開発全体としての生産性を向上させることが期される。加えて、バグの生じるパターン等を学習することによって、より効率的なバグの発見と修正を実現することが期待できよう。

コード生成 AI の技術的な方向性としては、基本的にはコード生成自動化の進化としてプログラミングの知識を有さない一般ユーザによるプログラミングの実現、完全な自動化プログラミングの実現、さらには自律的プログラミングの実現が大きく期待されている。もっとも、自動化の完全なる実現より、ソフトウエア開発の生産性向上への期待、すなわちコード生成 AI がソフトウエアの生産性向上に対して多くのインパクトをもたらすことに対して大きく期待されていることは、近時の調査研究でも明らかなようである。例えば、Keystone 調査報告[(17)]では、GitHub Copilot ユーザ 93 万人余りを対象とした調査によって、コード補完のサジェスチョンの 30％余りを利用して生産性が向上したこと、サーバの実装における比較対象実験によると GitHub Copilot を用いたグループの方が 55.8％速く実現できたこと、GitHub Copilot の利用と開発における生産性と満足度の間に強いプラスの相関関係がみられること、長期間にわたって利用することによってより大きなインパクトが生じること、経験の浅いプログラマの方に対して与えるインパクトが大きいこと[(18)]、等が明らかにされている。

また、コード生成 AI がソフトウエア開発にもたらす影響については、1000 人以上の当該分野の専門家に

（17）“Sea Change in Software Development: Economic and Productivity Analysis of the AI-Powered Developer Lifecycle”, Thomas Dohmke, GitHub; Marco Iansiti, Harvard Business School, Keystone.AI; Greg Richards Keystone.AI, arXiv:2306.15033 [econ.GN] June. 2023（2024 年 8 月 20 日最終閲覧）

（18）なお、GitHub の生成 AI 関係のレポジトリのうちもっとも多くを占めるトップ 20 のうち、9つは個人、2つはビッグテック企業、残り 9つはスタートアップ等の小組織によって運営されており、個人の開発者が極めて大きなインパクトを有している点は非常に興味深い。

対して、ソフトウエア開発におけるコード生成AIのインパクトと課題についてなされたGitLabによる調査研究[19]によって、日常的な開発業務のうちほぼ60％は、コード生成AI導入によって精度、効率性、生産性の向上が図られること、コード生成系AI導入の課題として、データのプライバシー保護、知的財産権、セキュリティが挙げられること、等が明らかにされている。もっとも、これらの調査研究は、いずれもコード生成AIのプラットフォームについて積極的に構築・提供を進めているGitHub（あるいは関連団体）がコミットしている調査研究である[20]ことには注意を要する。すなわち、コード生成AIが現実問題としてどの程度プログラムの生産性向上へ実効性があるのか、という評価については、若干バイアスが生じていることが否めない面はあり、未知数の部分があることも否定できないかもしれない。とはいえ、コード生成AIにより生成されたプログラムコードにバグが多いといった問題はあるものの、プログラム開発における生産性を向上させうることを指摘する見解は他にも少なくないものとみられる[21]。このようなことから、コード生成AIが今後のソフトウエア開発に拡大的に利用される方向へ進む可能性は極めて高いものと考えられる。

ところで、コード生成AIによって生成されるプログラムコードについてソフトウエア関連発明として特許法の下での発明として保護される可能性はあるのか、という問題についても若干検討を要するであろう。

先にも検討したように、現状の技術では、コード生成AIといえども、入力段階で人間が（出力として求める）プログラムの仕様等を何らかの形で特定して入力することが基本的に必要とされており、AIとして自律的にアルゴリズムを生成して、さらにそれに対応した特定のプログラムコードを自動的に生成・出力するといった段階にまでは至っていないものといえる。この点、完全自律的なコンテンツ生成を未だ行うものではない点はコンテンツ生成AIでも同様の状況にあるといえる。もちろん、入力されたプロンプトがかなり漠然とした内容であっても、それに対応して一定の処理を行うコードを生成･出力することもあり得るが、その生成過程とは、突き詰めると、入力されたプロンプトやコードの一部表記を基になされる（確率過程を基礎とする）推論の最適化に過ぎないものと考えられるのであって、アルゴリズム自体（という思想）をゼロから生成して、それを基にコードを生成しているものではないものと考えられる。このため、特許法におけるプログラム関連発明となり得るような創作が、コード生成AIによって生成されるという段階には至っている状況ではないと考えられる。

現行の特許法では、プログラムを物の発明として扱って、保護対象としているものの、あくまでも2条1項で定義される技術的思想としての「発明」を保護の対象としており、プログラムの技術思想を実現するために記述された個別具体的なコード自体を発明として保護対象とするものではないと解される。もちろん、生成されたコードの背景に何らかの技術的思想の存在を見出すことができて、さらに、その技術的思想が特許法の「発明」該当性を充たすものと評価しうる場合が生じることも否定することはできない。しかしながら、それは他の技術開発の場面で生成AIを用いて得られた回答から何らかの技術的思想が存していることを見出すような場合と何ら変わることはないといえるであろう。

したがって、コード生成AIと特許法を巡る問題については、他の技術分野の開発過程においてAIを活用した発明（特に化学や材料、医薬といった領域）の特許法における取扱いを巡る議論レベルと特段異なる

---

(19) GitLab,“The State of AI in Software Development.” Sep. 2023（https://www.tsoftglobal.com/wp-content/uploads/2023/11/GitLab-STATE-OF-AI.pdf）（2024年8月20日最終閲覧）

(20) 前者のKeystoneの調査研究は対象母集団がそもそもGitHub Copilotユーザであるし、後者の調査もGitHubの関連機関であるGitLabによるものである。

(21) 特定のソフトウエア開発企業を基にしたケーススタディから、ソフトウエア開発に生成AIを用いることについて、課題も少なくないが概ね肯定的な意義があるとの結論が得られたとする研究として、Mariana Coutinho, et al., The Role of Generative AI in Software Development Productivity: A Pilot Case Study, arXiv:2406.00560 [cs.SE], 1 Jun 2024,（2024年8月20日最終閲覧）。
企業サイドでも、例えば、IBMのように、生成AIがソフトウエア開発の生産性を向上させることを前提として、それを実現させるための要因を分析検討するものがみられる（Cole Stryker,“9 ways developer productivity gets a boost from generative AI”（March 6, 2024）（https://www.ibm.com/blog/developer-productivity/）（2024年8月20日最終閲覧））。

ものではないように考える。言い換えれば、コード生成 AI を利用することに伴って、ソフトウエア関連発明がとりわけ容易に創作しやすくなるであるとか、人間の創作活動を全く必要とすることなくソフトウエア関連発明を得ることができるといった理解は、少なくとも現時点での生成 AI を前提とする限りでは困難であるものと考える。

3．GitHub Copilot を巡る法的紛争の状況と検討

以下では、コード生成 AI と著作権に関連する法的紛争として現在も進行中である、アメリカにおける GitHub Copilot を巡る事例[(22)]についての検討を行う。

まず、GitHub は、2008 年に設立された、オープンソースモデルの下でソフトウエア開発のプラットフォームサービスを提供する事業者[(23)]である。GitHub は、Git と呼ばれるソフトウエアのヴァージョン管理ソフトウエアをツールとしてサイトにアクセスした上で利用可能なプラットフォームサービスを提供している。絶えず部分的なアップデートを繰り返すことが避けられないソフトウエア開発において、同時並行的に進行してゆくヴァージョン管理を開発者間で円滑に共有可能とすることが開発現場における大きな課題であるところ、Git は、円滑なヴァージョン管理とりわけ複数開発者間でのヴァージョンアップの管理をスムースにする機能で優れており、複数開発者によるコラボ開発における協調性を大きく向上させたことから、世界中のソフトウエア開発者間で極めて広く普及している[(24)]。GitHub ユーザとは、基本的にはプログラム開発者であって、自らが作成したプログラムコードを GitHub 上のレポジトリに公開して、他のプログラム開発者からのフィードバックを基に相互にコード開発を進める。プラットフォーム上に開設しているレポジトリにコード開発者が自ら開発したコードを公開して、各レポジトリに参加する他の開発者がさらに修正改良を加えたコードを公開して、開発を進めていくというイメージでの開発環境を提供している。したがって、ユーザは自ら作成したプログラムコードの著作権者として、GitHub 上のレポジトリにおいて、各種 OSS（オープンソースソフトウエア）のライセンスの下でコードを公開している。なお、GitHub の利用に際しては、利用規約に同意の上で登録して、アカウントを取得する必要がある[(25)]。

現在、法的紛争の対象になっている GitHub Copilot とは、自律的にコード生成をするものではなく、入力されたプロンプトに対応したコードの生成やコードの一部を補完するという、受動的（passive）にコード（を）補完・生成する AI であって、核となるコード生成に特化した学習モデルとしては Codex を用いている。Codex は OpenAI が構築したものであって、構築に際しての学習では、GitHub 各ユーザが GitHub 上のレポジトリにアップロードしているプログラムコードを学習データとして利用していた。

GitHub Copilot の公開・サービス提供の時系列としては、2021 年 6 月に Copilot を限定公開（開発は 2019 年から開始）して、2021 年 8 月に OpenAI は Codex を限定公開していたが、2022 年 6 月から GitHub Copilot のサービスを有料化した[(26)]。

Codex 及び Copilot としての利用時の態様としては、GitHub 上のレポジトリにあるプログラムコードを学習データとして利用していたものの、生成・出力されたコードにおいては、学習に用いられた各コードの

(22) 前掲注 3（DOE 1 et al. v. GitHub, Inc. et al, 4:22-cv-06823,（N.D. Cal.））
(23) もともとは独立した組織であったが、2018 年 10 月に Microsoft の子会社となっている。ただし活動については独立したものであることを Microsoft は強調しており（子会社化に際して “Microsoft Loves Open Source.” との見解を表明している）、その点は Git ユーザにおいても受容されてきていたとされる。
(24) Git ユーザは世界で 1 億人（2023 年 1 月の同社リリース）ともいわれており、情報系のエンジニア・研究者には極めて一般的に利用されているツールである。
(25) この利用規約上、レポジトリに公開したコードを学習データとして利用することまでも含まれるか否かも今回、原告主張の契約違反の一事項であることから、訴訟上は、利用規約やプライバシーポリシーの解釈も争点の一つとなっている。
(26) サブスクリプションによる利用の対価として 10 ドル / 月または 100 ドル / 年の有料制となっている。

著作権表示や提供するOSSライセンスについての記述が一切提示されていなかった(27)。このため、上記サービス有料化を契機として、GitHubユーザを中心として、OSSとして公開しているコードの事実上の「囲い込み」となることに対する強い反発と懸念が示されることとなって、2022年11月3日にGitHubで自らのプログラムコードを公開していた複数（2名）の著作権者（原告）がGitHub、OpenAI、Microsoftを被告としてカリフォルニア北部地区連邦地裁にクラスアクション（集団訴訟）を提起した。

原告の請求内容としては、差止請求として、原告が主張する（被告による）すべての違反行為の差止めを求めている。具体的には、学習データとして利用される過程で除去された前記情報（各コードの著作権表示や提供するOSSライセンスについての記述）がCopilotの出力に含まれるようにすることの確保を請求している。また、DMCA（デジタルミレニアム著作権法）違反行為（17 U.S.C. § 1203（b）(3)、17 U.S.C. § 1203（c）(2)、17 U.S.C. §1203（c）(3)）を根拠とした損害賠償請求、GitHubが原告のプログラムコードを学習データとして利用して構築したモデルを用いて、被告がサービスの有料化（サブスクリプション）によって利益を得ていることを根拠としてカリフォルニア州法による不当利得返還請求(28)等を行っている。

訴訟の進行状況(29)としては、2024年12月現在も連邦地裁に係属し、審理中である(30)が、2023年5月11日に連邦地裁決定（Order）が下され、一部の争点については却下されたうえで審理対象とする争点が絞り込まれた。

以下、上記の連邦地裁決定における主な争点に係る判断について検討する。

まず、原告主張の法的争点としては、権利管理情報に関するデジタルミレニアム著作権法（DMCA）の違反、ランハム法違反（ユーザによるコード提供行為に対する不正競争行為）(31)、ユーザのコード提供行為に対する州法レベルでの不正競争行為(32)、ユーザの提供するコードに係るOSSライセンス違反に伴う契約違反(33)、ユーザの提供するコードに含まれた情報を不正利用したことに係るプライバシー権侵害(34)、GitHubの利用条件違反(35)、といった事項を挙げている。

そのうち原告主張の中心となっている争点とは、Codex及びCopilotを構築するべく、GitHubのレポジトリで公開されている膨大なコードをCodex構築の学習データとして利用するに際して、原告が自らのプログラムコードを公開する際に適用している各種OSSライセンスに共通した、以下の3つの条件に違反し

---

(27) もっとも、現実的にみれば、原告主張のような形で、生成されたコードにこのような表示を行うこと自体が、そもそも非常に難しいようにも考えられる。

(28) なお、この点については、本件訴訟では、コードの学習利用による著作権侵害自体は主張しておらず、本請求は著作権法を根拠とすべきものであることから認められない、との判断を裁判所は早い段階で示している。

(29) 訴訟の主な動きについては、本件についてのサイト（https://githubcopilotlitigation.com/case-updates.html）で適宜更新されている。

(30) 時系列上の訴訟の流れについてはcourt listener（https://www.courtlistener.com/docket/65669506/doe-1-v-github-inc/）のサイトで適宜公開されている。なお、その後、原告は3名追加して参加しており、5名となっているようである。

(31) 連邦ランハム法1125条（15 U.S.C. § 1125）

(32) カリフォルニア州法（Unfair Competition law, Cal. Bus. & Prof. Code §§ 17200, et seq）

(33) Contract regarding the Suggested Licenses, GitHub's Privacy Statement

(34) カリフォルニア州消費者プライバシー法（California Consumer Privacy Act, "CCPA"）, Cal. Civ. Code § 1798.150

(35) GitHub's Terms of Service, Cal. Bus. & Prof. Code §§ 22575–22579, Cal. Civ. Code § 1798.150.
なお、GitHubの利用規約（The GitHub Terms of Service Effective date: November 16, 2020）を参考までに確認しておくと、
"D. User-Generated Content
4. License Grant to Us We need the legal right to do things like host Your Content, publish it, and share it. You grant us and our legal successors the right to store, archive, parse, and display Your Content, and make incidental copies, as necessary to provide the Service, including improving the Service over time. This license includes the right to do things like copy it to our database and make backups; show it to you and other users; parse it into a search index or **otherwise analyze it on our servers**; share it with other users; and perform it, in case Your Content is something like music or video. **This license does not grant GitHub the right to sell Your Content.** It also does not grant GitHub the right to otherwise distribute or use Your Content outside of our provision of the Service, except that as part of the right to archive Your Content, GitHub may permit our partners to store and archive Your Content in public repositories in connection with the GitHub Arctic Code Vault and GitHub Archive Program."（太字下線は筆者による）

て利用したという点である。

この3点を簡潔にまとめると、元のプログラムコードの著作権者を明らかにしていない点、著作権の表示を付さなかった点、適用されるOSSのライセンス条項を付加することなく利用した点である。したがって、本件訴訟の中心的争点とはOSSのライセンス違反を基礎とするものであって、著作権法の下での支分権侵害を争点とするものではない点には注意を要する。

もっとも、原告主張としては、Copilotは、GitHubのレポジトリ内にあるコードをそのまま利用して、あたかもCopilotが生成したように出力コードとしているとの主張も併せて行っている。この場合は、正当な権原なく、Copilotが原告のプログラムコードを複製していることを前提とするものになると考えられることから、著作権侵害の問題が別途争点化することも考えられなくもない。

また、そもそもGitHubのレポジトリにおけるプログラムコードの公開に際しては、13種類の「推奨される」OSSライセンスからコード作成者が選択して公開できる形となっている。このうちThe Creative Commons Zero v1.0 Universal及びthe Unlicenseの2つのライセンスにおいては、当該コードに係る著作権を一切主張しない内容とする条件となっているため、今回の訴訟の対象には含まれていない(36)。

このため、その解釈が問題となるOSSライセンスは、具体的には11のライセンス(37)である。このうち、本件原告2名の適用しているライセンスは、J. Doe 1（原告1）がMIT License , GNU General Public License version 3.0.、J. Doe 2（原告2）がMIT License; GNU General Public License version 3.0; GNU Affero General Public License version 3.0; The 3-Clause BSD License; Apache License 2.0. であって、これらのライセンスに共通した3つの条件とは、先にも簡単にまとめたように、当該OSSのライセンス条件下で提供されるコードを用いた二次的コード及び複製物の利用に際して、attribution to the owner of the Licensed Materials（“Attribution”）

“A. Definitions
6. The “Service” refers to the applications, software, products, and services provided by GitHub, including any Beta Previews. ”
下線部分（特に太線下線）の文言解釈からすれば、ユーザがGitHubにアップロード公開したコードを学習データとして解析利用する行為も利用規約の範囲内と解釈できる余地もありそうにも捉えられる。また、Copilotのサービス自体は上記利用規約の範囲を超えるようにも一見思われるが、“Service”の定義規定からすれば、その範囲に含まれる余地も否定はできないところである。
なお、あくまで参考ではあるが、本件訴訟では俎上に載っていないmoral rightsの取扱いについては、念のためGitHubの利用規約を確認すると以下のような規定がある。
“The GitHub Terms of Service Effective date: November 16, 2020
D. User-Generated Content
7. Moral Rights You retain all moral rights to Your Content that you upload, publish, or submit to any part of the Service, including the rights of integrity and attribution. However, you waive these rights and agree not to assert them against us, to enable us to reasonably exercise the rights granted in Section D.4, but not otherwise. To the extent this agreement is not enforceable by applicable law, you grant GitHub the rights we need to use Your Content without attribution and to make reasonable adaptations of Your Content as necessary to render the Website and provide the Service.”（下線は筆者による）
下線部のように、一応、不行使特約と理解される条項内容となっている。ただし、アメリカ著作権法の下でのMoral Rightsについては、その対象となる著作物が限定されていることから、ここで問題となるプログラムのコードについてはアメリカ著作権を前提とした請求という限りでは、実質的な意味はあまりないようにも考えられる。

(36) 原告クラスの定義そのものからも外れるものとなる。

(37) (1) Apache License 2.0（“Apache 2.0”）;
(2) GNU General Public License version 3（“GPL-3.0”）;
(3) MIT License（“MIT”）;
(4) The 2-Clause BSD License（“BSD 2”）;
(5) The 3-Clause BSD License（“BSD 3”）;
(6) Boost Software License（“BSL-1.0”）;
(7) Eclipse Public License 2.0（“EPL-2.0”）;
(8) GNU Affero General Public License version 3（“AGPL-3.0”）;
(9) GNU General Public License version 2（“GPL-2.0”）;
(10) GNU Lesser General Public License version 2.1（“LGPL-2.1”）;
(11) Mozilla Public License 2.0（“MPL-2.0”）.

（元のコードの権利者を明らかにすること）、inclusion of a copyright notice（“Copyright Notice”）（著作権が存することの告知）、inclusion of the applicable Suggested License’s text（“License Terms”）（適用されるライセンス条件（適用するOSS）の文面を含めること）、という各点となる。

また、DMCAに係る原告主張は以下のようなものである。DMCAにおけるCMI（Copyright Management Information）の定義としては、1202条（c）[38]に規定されているところ、被告が虚偽のCMIの提供、拡布等（17 U.S.C.§1202（a）[39]）を行ったとする原告主張については、裁判所は早い段階で採用しておらず、CMIの除去・改変等（17 U.S.C.§1202（b））[40]を根拠として、具体的には、被告が（b）（1）意図的にCMIを著作権者に無許諾で除去・改変する行為、（b）（2）CMIが著作権者に無許諾で除去・改変されたことを知りながらCMIを頒布または頒布のために輸入する行為、（b）（3）（著作権法の下での権利）侵害行為を誘引、可能化、容易化、隠ぺいされることについて知り、または1203条の下での民事上の救済に関しては、合理的にそのことを知る理由があるにもかかわらず、CMIが著作権者に無許諾で除去・改変されたことを知りながら、著作物、その複製物、レコードを頒布、頒布のための輸入をする行為、公衆への演奏等行為、を行ったとするものが主な原告の主張内容となっている。

上記の原告主張に対して、1202条（b）（1）及び（b）（3）では積極的な除去・改変を要するものであるため、被告行為は該当しないと被告は主張しているが、上記連邦地裁決定では、この被告主張を認めていない。他方、1202条（b）（2）に係る原告主張については、（あくまでもpleadingのレベルであるが）被告がCMIが除去・改変されたことを認識してプログラムコードを提供していたことにつき十分な根拠が示されているともいえないとして上記連邦地裁決定では認めていない[41]。

その他の原告主張に対する、上記連邦地裁決定における判断として、原告による契約違反の主張について

（38）17 U.S.C.§1202（c）
（1）the title and other information（著作権の表示を含む、当該著作物の権原その他の情報）
（2）the name of, and other identifying information about, the authors（当該著作物の著作者の名前その他の識別情報）
（3）the name of, and other identifying information about, the copyright owners（著作権の表示を含む、当該著作物の著作権者の名前その他の識別情報）
（6）terms and conditions for use of the work（当該著作物の利用に係る条件）
（7）identifying numbers or symbols referring to CMI or links to CMI.（著作権管理情報を示す、又は結びつける、番号又は記号）

（39）同条の条文は以下のようなものである。
（a）False Copyright Management Information.—No person shall knowingly and with the intent to induce, enable, facilitate, or conceal infringement— （1）provide copyright management information that is false, or （2）distribute or import for distribution copyright management information that is false.

（40）同条の条文は以下のようなものである。
（b）Removal or Alteration of Copyright Management Information.— No person shall, without the authority of the copyright owner or the law— （1）intentionally remove or alter any copyright management information, （2）distribute or import for distribution copyright management information knowing that the copyright management information has been removed or altered without authority of the copyright owner or the law, or （3）distribute, import for distribution, or publicly perform works, copies of works, or phonorecords, knowing that copyright management information has been removed or altered without authority of the copyright owner or the law,

（41）さらに、その後2024年1月3日に示された連邦地裁決定（https://storage.courtlistener.com/recap/gov.uscourts.cand.403220/gov.uscourts.cand.403220.195.0_1.pdf）では、この点について、1202条（b）（1）及び（b）（3）の下での主張は、著作物として保護される表現と同一の（identical）複製物からCMIが除去･改変されている場合であることを要するのであって、これに対して、原告主張は、Copilotが生成したコードは、原告の元のプログラムコードに修正がなされた表現であるとか、極めて類似しているものであるとか、重要でない部分が異なるバリエーションに過ぎないとか、同じアルゴリズムを実現するものである、機能的に等しいものであるといった主張をしているが、十分に説得的なものではないとして、被告主張を認め、原告主張を斥けた。ただし、裁判所は原告主張を修正・補足する機会（Leave to amend）を与えており、2024年3月時点では、この決定に対する原告主張の修正やさらに被告による主張が提出されている状況にあったが、その後の修正・補足を踏まえて。2024年6月24日の連邦地裁決定（https://storage.courtlistener.com/recap/gov.uscourts.cand.403220/gov.uscourts.cand.403220.253.0_1.pdf）では、1202条（b）（2）の下でも、著作物として保護される表現と同一であることの要件（identicity requirement）を充足する必要があると解されるが、原告主張はこれを充たしていないとする判断を示した。CMIの除去･改変に対する法的規制について、CMIが付加されている著作物の同一性が問題となるという論点は非常に興味深いところであるが、2024年6月24日の連

は審理対象とすることを認める判断を示した(42)。また、ランハム法、カリフォルニア州法の下での不正競争、プライバシー権侵害、GitHub のプライバシーポリシー、利用規約違反等による原告主張については認めなかった。

以上を整理すると、上記連邦地裁決定としては、原告主張のうち、DMCA§1202（b）（1）及び（b）（3）を基にした主張、契約違反（GitHub のプライバシーポリシー、利用規約に係る内容以外の OSS ライセンス違反）を基にした主張を審理対象とすることを認めたといえる。それ以外の原告主張については却下（dismiss）した。

4．コンテンツ生成 AI とコード生成 AI の差異ー現象面と法的側面

次に、生成 AI のうち、テキスト、画像や動画、音楽といったコンテンツを中心とした生成物を出力する AI（以下、便宜的にコンテンツ生成 AI とする）とプログラムコードを生成する AI（コード生成 AI）の間における差異としては、いかなる点が見出せるのか、現象面と法的側面に分けて考察する。

まず、現象面としては、両者の間では、生成に用いられるモデルの種類が異なることが多いと考えられる。現状では、コード生成には基本的に大規模言語モデル（LLM）が用いられることが一般的であると考えられるが、コンテンツ生成の場合、近時は先に概観したように拡散モデルが用いられることが主流となりつつあるようである(43)。また、プログラムコードの場合、各プログラミング言語の文法に従った形でコードとしての表現が成立していることを考えると、コード生成の場合、既存のプログラムコードの十分な蓄積を学習データとして活用することによって、より実用性の高いコード生成が高い確率で実現されることが期待できるかもしれない(44)。このため、実用面として利用に直結するという観点からすれば、コード生成 AI の方がコンテンツ生成 AI に比べて、急速な発展普及をする可能性も高くなることが期待できる。

法的側面として、コード生成 AI の開発と利用の局面における著作権法上の課題は、コンテンツ生成 AI と基本的には同質であると考えられる。現段階における具体的課題としては、AI 開発段階（特に学習モデル構築段階）において著作物を学習データとして利用する行為と著作権侵害の成否、AI 利用段階での生成行為についての（既存の著作物に係る）著作権侵害の成否といったものが挙げられると考えられる。

もっとも、コード生成の場合、プログラミング言語の文法や技術的な制約によって、特定の機能を実現するプログラムコードとしての表現の記述様式の幅が必然的に小さくなる状況も少なくないかもしれない。このような場合、類似あるいは同様の内容をもった指示やプロンプトが入力されることによって、それを基に推論がなされて生成されるコードとしても、学習モデルを構築する際に用いられたプログラムコード自体の記述に類似したコードが生成･出力される可能性が高くなることも考えられ、このような現象が生じる場合、学習データがそのまま出力されたのか、あくまでも推論による結果として類似する記述のコードが生成されたのか、という両者の間を峻別することが一層困難となろう。そして、この点を契機として著作権侵害の成否に係る法的紛争が生じやすくなることが想定される。実際に、先の GitHub Copilot の事例における原告主張でも原告プログラムコードとほぼ同一のプログラムコードが生成されたとする主張がみられる。他方、生成・出力されたプログラムコード自体が、学習データに用いられたプログラムコードに類似していると評

邦地裁決定については、さらに、被告が同決定における 1202 条（b）の解釈を不服として、控訴裁判所への中間上訴（interlocutory appeal）を求め、2024 年 9 月 27 日の連邦地裁決定で、これが認められて、控訴審での審理がなされることになったという（この点についての詳細な解説として、宮下佳之・生成 AI の開発過程での学習と著作権管理情報の除去又は改変に伴う責任・SLN（一般財団法人ソフトウエア情報センター）179 号を参照）。

(42) 具体的内容についての十分な特定がなされているとはいえないものの、原告主張はプリーディング（訴答）には足るものとして、被告の却下申立ては認めなかった。

(43) もっとも、コンテンツ系について LLM モデルを用いて生成することも可能である。

(44) ただし、現状の機械学習・深層学習をベースとした AI では、プログラミング言語の文法そのものを「理解」してコードを生成するという処理を行うことは困難と考えられ、その意味では、優れたプログラマによるコーディングについてのノウハウを反映したコード生成が可能となるということを直ちに意味するものではないのかもしれない。

価されるとしても、当該学習データに用いられたプログラムコード自体について、機能実現の観点からの制約が存在するものとして、そもそも著作物性が否定されるような場合も少なくないとも考えられる。この点は、表現を記述する振れ幅が一般的には広いものと考えられるコンテンツ系の場合と状況を大きく異にする点であるといえるであろう。

また、上記で検討したGitHub Copilotを巡る事例から得られる示唆として、コード生成AIの場合、基本的にOSSライセンスの下で公開されているプログラムコードが学習データの多くを占めるものと考えられる(45)ことから、学習データに用いられうるプログラムコードの利用自体については、OSSライセンスの規定する許諾条件の下での枠内である限り元々広く許されているという前提がある。このため、学習段階での他人の著作物利用の問題を検討するに際して、OSSライセンスの解釈という問題が一段加わることが、コンテンツ生成AIの状況と比べて大きく異なる点であるものと考えられる(46)。

OSSライセンスの下で提供されているプログラムコードについては、具体的には、以下のような事項が問題となり得るものといえる。

第一に、モデル構築のための学習データとして利用する行為がOSSライセンスの下での利用条件を充たす行為に該当するのか、という点である。この点は、それぞれのOSSライセンスにおける条項の解釈問題となるであろう。

第二に、仮に学習データとして利用する行為がOSSライセンスの下での利用条件に服する行為に該当すると解されるとするのであれば、OSSライセンスで規定された効果が発生することによって、学習モデルを用いて生成されたコードに対しても当該OSSライセンスの効果が及ぶものと解されるのか、という点が挙げられる。

生成されたプログラムコードに対しても、学習データに用いられたプログラムコードに係る著作権の効力が及ぶと理解するのであれば、第一の点について学習データとしての利用をもってOSSライセンスの下での利用条件を充たすとする立場の下では、OSSライセンスの効力は生成されたプログラムコードに対しても及ぶものと解されざるを得ないものと考えられる。逆に効力は及ばないと理解するのであれば、OSSライセンスの効力は生成されたコードに対しては基本的には及ばないことになろう。すると、このような場合、生成されたコードはそもそも著作権法上の著作物と評価しうるのか、仮に著作物であるとすれば著作者はどのように捉えるべきであろうか、という問題が付随的に発生することも考えられるであろう。

第三に、生成されたプログラムコードに対してもOSSライセンスの効果が及ぶと解される場合には、生成されたコードの利用に際してOSSライセンスの下で要求されている事項の履行を確保させることが現実的に可能であるのか、という点が挙げられる。この点で、生成されたコードについても、元のコードのOSSライセンス条件の下で提供するといった取扱いを求めること自体は可能と考えられる。他方で、著作権に係る表示やOSSライセンス文書の付加等を当該生成されたプログラムコードを生成したAIの学習に関わったすべてのOSSライセンスについて行うことを求めることは非現実的であるようにも考えられる。例えば、コード生成に影響を及ぼしたと考えられる学習データとなったプログラムコードすべてについて、それらに係る著作権表示やライセンス条項の付加をする義務が法的にも生じるとすれば、おそらく膨大なものとなることが容易に想定できるのであって、現実的には履行困難であるように考えられる。

---

(45) 特定の企業が自社製品として、あるいは主に自社利用のために開発するようなprorietaryなコードは、そもそもソースコード自体が公開されないことが一般的と考えられるために学習データとして利用すること自体が困難であることから、コード生成AIにおいては、実質的にはOSSライセンスの下で公開されているコードを中心として学習データに用いて構築されたモデルをベースとしたコード生成が前提となるであろう。

(46) もちろん、コンテンツ系においても、creative commons等のOSSライセンス条件で一定の利用行為を広く許諾しているものもみられるといえるから、相対的な差異に過ぎないとみることもできるかもしれない。

５．日本法の下での検討

以上のGitHub Copilotの事例検討やコード生成AIと著作権法に係る課題も踏まえて、アメリカにおけるGitHub Copilotに係る事例を、日本法の下に置き換えて考察した場合に考えられる問題について、以下、検討を試みる。

まず、学習モデル構築段階では、OSSライセンスの解釈問題として、レポジトリ等で公開されているコードを学習データとして利用する行為がOSSライセンスに違反するのか、という問題が挙げられる。この問題は、先にも述べたように、基本的には個別のOSSのライセンス条項をどのように解釈するのかということ次第で結論が変わってくるものといえる。

多くのOSSライセンスが利用条件としてuse、copy、modifyといった行為を挙げており、とりわけ複製については、OSSライセンスのほとんどが利用条件に含めて規定しているものと考えられ、他方で既存のコードを学習データとして利用するための処理を行うに際しては、少なくとも複製は不可避的に生じうるものと考えられる。このため、一つの考え方としては、多くのOSSライセンスの下で提供されるコードであっても、学習データとしての利用に際しての複製行為を接点としてOSSライセンスの下での利用関係に入るものと解しうる余地があるものと考えられる。

他方で、異なる解釈も考えられる。すなわち、多くのOSSライセンスで列挙されているuse、copy、modifyといった上記の利用行為の概念について、当該プログラムコード自体の有する機能を実現する目的という文脈にあくまで限定してuse、copy、modify等の行為を行う範囲に留まるという限定的な解釈をするのであれば、学習データとして利用する行為とは、あくまで生成AIのモデル構築を目的としている行為であると考えられるのであって、OSSライセンスの利用条件には、学習データとしての利用行為はそもそも含まれないという解釈も成り立つ余地は否定できないかもしれない。そもそもOSSライセンスとは、プログラムコードを公開して、自由な改変を許容することによって、より優れたプログラムコードの開発を促進し、その成果を広くコミュニティ間で共有するという思想を出発点としていたものであることに回帰するのであれば、自然人であるプログラマがプログラムコードの果たす機能を実装・実現あるいは改良することを大前提としているものと考えることも可能ではあるといえるのであって、機械学習・深層学習のためのモデル構築のための「データ」として利用することを目的とする場合は、OSSライセンスの本来の前提には該当しないものと解することも不可能ではないようにも考えられる。

前者の解釈を前提とするのであれば、各OSSライセンスの法的効果は学習モデル構築の段階における行為に対して生じる可能性はあると考えられ、その結果、著作者、著作権等の表示義務、OSSライセンス条項の付加義務といった各OSSライセンスで規定された義務を学習モデル構築者も負うことになりうるという結論も導き出せそうでもある。仮にそうだとすると、学習モデル構築者は、これらの義務を履行しない限り契約違反を構成することとなって、さらには、そのような学習モデルを用いて生成されたプログラムコードの利用権原も失われるものと解されるのか、というところまで至る問題が想定されることになろう。このため、いずれの方向性の解釈とすべきかという問題は、コード生成AIの利用の帰趨にまで大きな影響を及ぼしうる大きな問題であるともいえるかもしれない。

また、後者のような解釈が可能であるとすれば、学習段階での複製についてはOSSライセンスの利用条件に入らないと解される余地が生じるため、その法的効果も及ばないということになりそうである。しかし、逆にそうなると、学習段階でのプログラムコードを複製する行為については、プログラムコードの著作権者から別途許諾を受けることを要するのではないのかという結論に至ることも考えられうるであろう。

もっとも、日本法の場合、この点については学習データとしての著作物利用に係る著作権法の権利制限規定が作用する余地があるため、各OSSライセンスの内容如何にかかわらず、学習段階での行為自体が著作権侵害を構成しない可能性があるものと考えられる。このため、学習段階においてOSSライセンスの法的効果が及ぶという先の前者の立場を仮に前提としても、そもそも権利制限規定の対象と評価できる余地も

あって、OSSライセンスの規定する利用条件に関わらず、モデル構築段階での学習のためのプログラムコードの利用が許容されうるといえる。このような場合には、仮に学習モデル構築者がOSSライセンスの課している義務を履行せず、形式的にOSSライセンス違反を構成することとなっても、著作権法上は当該コードについての著作権侵害の法的効果は生じないということになりそうである。

ところで、著作権法の下での権利制限対象となりうるとしても、適用されうる権利制限規定としては具体的にどの規定となるものと解されるであろうか。

まず、法30条の4該当性を巡っては、非享受利用目的（当該著作物に表現された思想又は感情を自ら享受し又は他人に享受させることを目的としない場合）に当たるのか、という問題が認識できる。

プログラム著作物についての「享受」該当性については、当該プログラムを実行することを通じて、その機能に関する効用を得ることに向けられた行為であるかという観点から判断すべきとする見解(47)がみられる。なお、この立場では、リバースエンジニアリング（プログラムの調査解析を目的とする利用）については「非享受」になるものと評価している(48)。いわば、上記の見解とは、リバースエンジニアリングはプログラムの機能に関する効用を享受することに向けられたものではないとする理解(49)を基にしているのであって、これは一般的には受容されやすい考え方といえる反面で、プログラムの解析を行って、その背後にある技術思想を獲得して、開発に資するという意味では、プログラムの表現を通じて反映されている機能に関する効用を「享受」するという側面が全くないとは言い切れないようにも考えられるところである。すなわち、リバースエンジニアリングの場合、ハードウエアを動作させるためのバイナリやマシン語等のそもそも人間の自然言語からかけ離れた表現からプログラムコードの記述やアルゴリズムを読み取るという作業過程であることと対比してみると、プログラムコードに化体された表現上の特徴を抽出した上で学習モデルを構築して、それを用いて、AIへの入力（プロンプト）に対して最適解を推論して生成されたコードを出力するという過程についても、コードに表現された思想・感情についての享受目的が全くないとまで評価できるのかという点については若干疑問も生じるところである。

もっとも、この点は、表現と機能が不可分な関係性にあるプログラム著作物固有の性質から生じる特徴が多分に影響しているともいえるかもしれない。すなわち、プログラムコードに体現された表現の特徴利用と表現によって実現される機能利用を完全に分離すること自体がそもそも非常に困難という側面は否定できないと考えられる。プログラム著作物における上記のような特徴を踏まえつつ、リバースエンジニアリングを「非享受」と解する上記のような解釈論を基に捉えるのであれば、コード生成AIモデルの構築のための学習段階における既存コードの利用行為についても、コード表現の特徴抽出にとどまるものに過ぎないとも評価しうる余地はあると考えられるのであって、「非享受」目的での行為として解することも合理性を失するものではないと評価できるのかもしれない。

加えて、画像、音楽等のコンテンツ系については学習段階における学習データとしての利用行為である限り「非享受」利用と解する余地があると一般的に解されている(50)ところであって、コンテンツ系と比べて、プログラムコードの学習段階における利用が表現享受目的としての性質がとりわけ高いとは必ずしも評価し得ないようにも考えられる。

他方、コード生成AIにおける学習過程については、法30条の4の例示類型とされる2号の情報解析に文理上も該当するものと解する余地もあるといえる。情報解析については、「多数の著作物その他の大量の情報から、当該情報を構成する言語、音、影像その他の要素に係る情報を抽出し、比較、分類その他の解析

(47) 加戸守行『著作権法逐条講義（七訂新版）』282 － 283頁（公益社団法人 著作権情報センター、2021）
(48) 加戸・前掲注47・283頁
(49) 加戸・前掲注47・283頁
(50) 法30条の4の立法段階でも、むしろこの点を前提として立法されたものといえるであろう。加戸・前掲注47・284頁等

を行うこと」とされており、立法の趣旨としても、著作物の種類にかかわらず、深層学習における学習データとしての複製・記録等の行為については、2号該当性を肯定する余地は十分にあるものと解される(51)。

コード生成AIの学習段階におけるプログラム著作物の利用については、これまで検討したようにリバースエンジニアリングと完全に同質とは捉えられない特徴もみられるように考えられることから、コード表現の非享受目的と享受目的が併存するという考え方もありうるであろう。例えば、文化審議会著作権分科会法制問題小委員会から公表された資料(52)では、非享受目的以外の目的が併存する場合には、法30条の4の適用は否定され、法47条の5の適用が問題となる旨の見解がでている(53)。

そこで、コード生成AIの学習段階の法47条の5該当性について、次に検討を行う。コード生成AIの学習段階としては、同条に列挙されている類型のうち、2号に規定される情報解析には該当する余地があるように考えられる。ただし、以下の要件の充足が前提となるため、その充足可能性について検討を要する。

「電子計算機を用いた情報処理により新たな知見又は情報を創出することによつて著作物の利用の促進に資する行為」該当性要件については、コード生成AIの学習行為が充足することについてはあまり疑義が生じないものと評価できるものと考えられる。

公衆提供等著作物該当性要件については、先の事例のようにGitHubのレポジトリ上にアップロードされて公開されているコードであれば、基本的には充足しうるものと考えられる。また、アップロードされているのみで、未だアクセスあるいはダウンロードされていない状態にあるプログラムコードであっても、少なくとも送信可能化はされているものと評価しうるであろう。

目的上必要と認められる限度という要件について、コード生成AIのモデル構築のための解析・提供としての利用であれば､少なくとも同条2号の目的には合致しているものと解されるのではないかと考えられる。なお、コード生成AIがもたらしうるプログラム開発の生産性向上等々の利点を捉えれば同条3号の目的に合致しているものと評価することもできなくないとも考えられるが3号では政令指定を前提としていることから、現行法の状況を前提とする限りは該当性を肯定することは差し当たり難しいであろう。

付随性要件については、各号の利用行為との間に主従関係が必要(54)とされており、学習モデル構築のための解析・提供を同条2号に該当する利用行為に相当するものと解した上で、当該行為に付随してなされうる行為として捉えるのであれば、当該要件充足も肯定しうる余地はあるのではないかと考えられる。

軽微利用性要件については、占める割合、量、表示精度等の外形的要素を基に判断するものとされており、公共性は判断要素外とされている。コード生成AIの学習過程の場合は、基本的には学習データとして、各プログラムコードの表現全体を用いて学習段階で用いられなくては、学習過程としての意味がほとんどないものと考えられ、軽微利用と評価することは困難ではないかと考えられる。

以上の検討からすれば、主として軽微性要件の充足が困難である以上、法47条の5該当性を肯定することは難しいものと評価される。

このように、コード生成AIの学習段階については、仮に法30条の4該当性を否定した上で法47条の5の該当性を肯定することは、現行規定の解釈論として捉える限りは困難であるように考えられる。したがっ

(51) 加戸・前掲注47・285 − 286頁。また、3号該当性も併せて検討の余地はありそうである。もっとも、3号については、人の知覚による認識を伴わない場合を前提としているものと解され（加戸・前掲注47・286頁)、モデル構築に際しての学習段階においては、開発者による認識を全く伴わないと評価することにはやや無理があるように考えられる。今後、学習段階についても完全なる自動化がなされるようなことが生じるのであれば、3号該当性についても検討する意義が生じてくるかもしれない。

(52) 文化審議会著作権分科会法制問題小委員会「AIと著作権に関する考え方について」（令和6年3月15日）（https://www.bunka.go.jp/seisaku/bunkashingikai/chosakuken/pdf/94037901_01.pdf)

(53) 文化審議会著作権分科会法制問題小委員会・前掲注52。そもそも、この目的の「併存」という考え方自体について、私見としては疑問が生じなくもなく、同資料の例示する目的併存の例については、私見としては享受目的ありと評価しうる余地があるようにも理解できると考える。

(54) 加戸・前掲注47・417頁

て、解釈論としては、コード生成 AI の学習段階における既存のプログラムコード利用については法 30 条の 4（特に 2 号）該当性を根拠として、原則として権利制限対象となりうるものと解することが妥当と考える。

以上より、コード生成 AI のために用いられる学習モデルを構築するために OSS ライセンスで公開等されているコードを利用すること自体については、日本法の下では、原則として著作権法の権利制限類型（法 30 条の 4 第 2 号）に該当するものと解される余地はあるものと考えられるであろう。

OSS ライセンスの下で提供されているプログラムコードを学習データとして利用する行為自体については、具体的な各 OSS ライセンスの趣旨には必ずしも整合しうるものではなく、むしろその趣旨に反する部分があることは否定できないであろう。他方で、OSS ライセンス自体について、一般的な意味での著作権ライセンス契約としての性質よりも、プログラムコードを広く利用することを受容した上で、その利用に際しての「約束事」を定めているという性質が強い傾向はみられるのであって、加えて、日本法の下で権利制限対象となる行為と評価しうる以上は、仮に OSS ライセンスが本来前提としている利用態様に適合しない行為として評価されるとしても著作権の効力が及ばないことは当然ともいえる。

このため、GitHub Copilot を巡るアメリカにおける事案のような場合、少なくとも日本法の下では学習段階における OSS ライセンス違反による効果として著作権侵害の問題は生じにくいように考えられる。とはいえ、著作権法に係る別系統の問題として、権利管理情報の取扱いの問題は残るのかもしれない。

具体的には、OSS ライセンスの下で提供されているプログラムコードを基に一部改変等によって得られたプログラムコードの提供に際して記載することが要求される著作権者等の表示が著作権法上の権利管理情報に該当するのか、という点であって、もし権利管理情報に該当するとした場合、学習段階で元のプログラムコードからこれらの情報を取り除いて学習モデル構築の段階で利用する行為については、みなし侵害（法 113 条 8 項 2 号[55]）を構成するのか、という問題である[56]。

OSS ライセンスによって表示等が課される情報についての権利管理情報該当性について、日本法の下では、まず権利管理情報について、定義規定である法 2 条 1 項 22 号では、「著作権等」に関する情報であって、イからハまでのいずれかに該当するもののうち、電磁的方法により著作物、実演、レコード又は放送若しくは有線放送に係る音若しくは影像とともに記録媒体に記録され、又は送信されるもの（著作物等の利用状況の把握、著作物等の利用の許諾に係る事務処理その他の著作権等の管理（電子計算機によるものに限る。）に用いられていないものを除く。）と規定されている。イからハについては、「**イ**　著作物等、著作権等を有する者その他政令で定める事項を特定する情報」については、具体的には著作物名、著作者名、著作権者名等が該当し、「**ロ**　著作物等の利用を許諾する場合の利用方法及び条件に関する情報」については、利用許諾の形態、対価等の情報等が該当し、「**ハ**　他の情報と照合することによりイ又はロに掲げる事項を特定することができることとなる情報」については、上記イ及びロの情報をデータベースで検索可能とするコード情報等が該当するものと解されている[57]。したがって、先に検討した GitHub Copilot の事案でも問題となっている CMI のうち、元のコードの著作権者の表示、適用されるライセンス条件（適用する OSS）の文面については、日本法の下での権利管理情報に該当するものと解される。

「故意に」の要件については、権利管理情報の存在確認義務を課するものではないとされる[58]。とはいえ、

---

(55) 同条では、権利管理情報を故意に除去し、又は改変する行為（記録又は送信の方式の変換に伴う技術的な制約による場合その他の著作物又は実演等の利用の目的及び態様に照らしやむを得ないと認められる場合を除く。）をもって、当該著作者人格権、著作権、出版権、実演家人格権又は著作隣接権を侵害する行為とみなすと規定している。

(56) 基本的には OSS ライセンスの下で提供されているプログラムコードについては、適用する OSS ライセンスの付加や著作権の表示等を併せて行っていることを前提とするが、この点は、必ずしも OSS ライセンスの内容として課される義務に関わらず、プログラムコードがその提供段階で著作権者等の表示を行っている場合には一般的に生じうる問題とも考えられる。

(57) 加戸・前掲注 47・70 － 71 頁

(58) 加戸・前掲注 47・868 頁

OSSライセンスの下で提供されているプログラムコードの場合、基本的にはプログラムコードと共に、これらの表示や付加が明確になされているものが多いと考えられることから、学習コードとして利用するに際して、これらの情報を除去する行為については、故意性を肯定しうる場合も少なくないものと評価できよう。

このような前提の下で、学習データとしての利用行為自体については法30条の4柱書の権利制限規定の対象となるものと解釈しうる場合については、当該行為（権利制限規定によって著作権を侵害する行為と評価されない行為、すなわち学習データとして利用する行為）に伴って権利管理情報を除去する行為について、直ちに法113条8項2号（カッコ書き）の「やむを得ない」に該当するものと解することができるのであろうか。換言すれば、法30条の4等の権利制限規定該当性を根拠として、当該行為に付随してなされる権利管理情報の除去等の行為について、直ちに「やむを得ない除去」として解することができるのかという問題が浮かび上がってくる。この点については、原則論としては、必ずしもそのように直ちに解することはできないものであって、除去行為が権利制限対象行為に付随して行われる行為であるか否かという問題はみなし侵害行為の評価の問題とは基本的に別途判断されるべき問題であるようにも考えられる。

もっとも、同条2号の趣旨とは、権利侵害のおそれを生じさせる行為を捉える趣旨の規定(59)とされていることからすれば、そもそも権利制限対象行為に伴って行われる除去行為である限りは、権利侵害のおそれを生じさせない行為であると評価できる余地はあるのかもしれない。

また、異なる観点からは、あくまで学習モデルの構築という目的を実現するためのデータとして供されるに留まるということを前提とするのであれば、権利管理情報を除去後、当該著作物の表現そのものはモデル構築に伴って「消化」されてしまって、権利管理情報の除去行為がなされて以降、当該表現自体が更なる利用に供される余地はもはや生じないから権利侵害のおそれもないものとして捉えることができるものと考える余地もあるかもしれない。しかしながら、この点についてコード生成AIでは、学習段階に引き続いて生成段階が生じることを当然の前提としている以上、学習段階のみならず生成段階で権利侵害を惹起するおそれが生じるのか否かという点までも考慮した上で、「やむを得ない除去」に該当するかについて判断をする必要があるものと考える。その上で、新たな生成物を生成する段階をもって著作権侵害を構成しうると評価する余地も払拭できないものと解されている現状(60)を考慮すると、OSSライセンスの下で提供されているプログラムコードのような場合については、原則論として、むしろ「やむを得ない」ものと直ちに評価しうる場合の方が少ないのではないかとも考えられるところである。

以上の検討からすれば、基本的には、非享受目的の学習段階での利用であることを直接の根拠とするだけでは、権利管理情報の除去行為のみなし侵害該当性を直ちに否定する結論を導出できるものでもないように考えられる。

このため、法30条の4等の権利制限規定の対象に該当するか否かは別の問題として、学習データとして利用されうるあらゆる著作物が権利管理情報を付加して提供されることが一般化した場合には、学習過程における当該権利管理情報の除去がみなし侵害を構成しうることが、法30条の4等の権利制限規定によって導き出される効果とは別途、実質的に学習データとしての著作物利用に際してのボトルネックとなることが懸念される事態も考えられるのである(61)。

逆の見方をするのであれば、権利管理情報の除去行為についてみなし侵害が否定される領域（あるいはそれを実現する要件論）が、より明確にされない限り、学習データとして利用されうる、あらゆる著作物に対して権利管理情報を付加することで、法30条の4等の権利制限規定の効果とは別の作用として、多様な著作物を学習データとして利用されることを阻止できる可能性も無視し得ないことが指摘できる。もちろん、

(59) 加戸・前掲注47・868頁
(60) 文化審議会著作権分科会法制問題小委員会・前掲注52・34頁
(61) この問題は コンテンツ系も含めた今後の課題といえるのかもしれない。

先に検討した連邦地裁決定のように、日本法においても、権利管理情報の除去の前後で著作物としての表現の同一性を要するという解釈が成り立つのであれば、まさにアメリカの裁判例と同様に同一性要件の充足が争点ともなり得るであろう。

次に、生成段階の問題として、コード生成AIによって生成・出力されたコードについて、OSSライセンスの法的効果が及ぶか否かの問題について検討する。この問題は、先に示したように、生成されたコードに対して学習に用いられたコードの著作権の効力が及ぶものと解するのか否かにあくまで依存する問題といえるのであって、コンテンツ系も含めて生成AIと著作権法を巡って一般的に議論されている問題と同質であってコード生成AI固有の課題ではない。

まず、モデルの構築に用いられたコードに近い表現を有するプログラムコードが生成・出力される可能性が一定程度存することは否定できない(62)といえることから、著作権侵害成否の議論となる。とりわけ依拠性についてどのように評価するのかという考え方が侵害の成否の範囲を決定付ける上でもっとも大きな問題と考えられる。コンテンツとプログラムコードのいずれもこの点は同じであろう。

この点、文化審議会著作権分科会法制問題小委員会のペーパーで示された考え方(63)を基にすると、プログラムコードが学習に用いられていたことをもって、コード生成AIの利用者の認識に関わらず、依拠性ありと評価しうる余地があるものといえる。仮にもこの理解を前提とするのであれば(64)、ユーザが認識していなかったコードを基に構築された学習モデルを利用して、コード生成をした場合、たまたま、学習データと類似性のあるプログラムコードが生成された場合であっても、原則として著作権の効力は及びうるということになる。

もっとも、当該学習コードがあるOSSライセンスの下で提供されている場合に、生成・出力されたプログラムコードを利用する行為については、OSSライセンス所定の条件を具備する態様をとる限りの範囲では、著作権侵害を構成することにはならないといえるであろう。したがって、この場合にはユーザレベルで、生成されたプログラムコード自体の複製等の利用を継続して行うこと自体は可能であると考えられる(65)。これらを踏まえると、仮に上記の立場を前提として、コード生成段階で類似（二次的著作物）・依拠を充足すると評価されたとしても、生成されたプログラムコード自体の利用等については、著作権侵害が問題となっている元のプログラムコードが適用しているOSSライセンスの利用条件を充足する形で行われている限りにおいては、その利用行為自体は妨げられないという結論になるのかもしれない。

---

(62) 生成を指示するプロンプト自体が（学習コードの下にある指令内容と）同様あるいは近似した指示内容を含むことが少なくないと考えられる。この点はコンテンツ系の場合もおそらく本質的には同じであろう。モデル構築のための学習に用いられたプログラムコードに同一・類似と評価される場合のみならず、二次的著作物として元のプログラムコードの著作権の効力は及ぶことになるから、一部改変された形で翻案物と評価されうるプログラムコードが生成される場合も同様に考慮する必要があろう。

(63) 文化審議会著作権分科会法制問題小委員会・前掲注52
「② AI利用者が既存の著作物を認識していなかったが、AI学習用データに当該著作物が含まれる場合
✓　AI利用者が既存の著作物（その表現内容）を認識していなかったが、当該生成AIの開発・学習段階で当該著作物を学習していた場合については、客観的に当該著作物へのアクセスがあったと認められることから、当該生成AIを利用し、当該著作物に類似した生成物が生成された場合は、通常、依拠性があったと推認され、著作権侵害になりうると考えられる。
✓　ただし、当該生成AIについて、開発・学習段階において学習に用いられた著作物の創作的表現が、生成・利用段階において生成されることはないといえるような技術的な措置が講じられているといえる場合もあり得る。このような技術的な措置が講じられていること等の事情から、当該生成AIにおいて、学習に用いられた著作物の創作的表現が、生成・利用段階において利用されていないと法的に評価できる場合には、AI利用者において当該評価を基礎づける事情を主張・立証することにより、当該生成AIの開発・学習段階で既存の著作物を学習していた場合であっても、依拠性がないと判断される場合はあり得ると考えられる。
✓ なお、生成AIの開発・学習段階で既存の著作物を学習していた場合において、AI利用者が著作権侵害を問われた場合、後掲（2）キのとおり、当該生成AIを開発した事業者においても、著作権侵害の規範的な主体として責任を負う場合があることについては留意が必要である。」（同上・34頁）

(64) この考え方に対しては、私見としては大いに疑問が生じるところである。

(65) コンテンツ系の場合、この点で結論の帰趨としてはOSSライセンスの効果が及ばないという点でかなり異なることが多いかもしれない。

ただ、OSSライセンスの効果として、当該コードについても同一のOSSライセンスの下で第三者に無償で利用に供することを要求している条項を有するOSSライセンスがあることから、そのようなOSSライセンスの下で提供されているコードが学習に用いられているコード生成AIによって、生成されたコードを商業的に利用するに際して、当該プログラムコードに特定の営業秘密等が組み入れられているような状況にある場合には、当該プログラムコードに「滲み込んでいる」営業秘密等の情報について公に流出しうるリスクが併せて発生することも懸念される。また、著作物であることの表示義務やOSSライセンスの付加義務といった法的義務は、OSSライセンスの下で要求されている限りはいずれにせよ生じうるであろう。

他方、生成されたプログラムコードに対しては学習に用いられたコードの著作権の効力は及ばない、すなわち、原則として依拠性は否定されうると理解した場合には、生成されたプログラムコードに対してOSSライセンスの効果は波及しないものと解される。このような場合には、生成されたプログラムコードは著作物として評価しうるのかという問題が次に考えられる。私見としては、著作物としては評価し得ないものが少なくないと考える(66)。そして、このように著作物性を肯定し得ないプログラムコードについては、当然ながら著作者も存在しないことになると考えられる。もちろん、プロンプトや入力指示の段階での創作的寄与が相応に認められることで著作物性が肯定され、生成させるための入力等を行ったAI利用者の著作物と評価しうる場合もありえなくはない。ただ、いずれの場合も、生成されたプログラムコードについては、学習データとなったコードが適用しているOSSライセンスの「縛り」からはもはや逃れられることになるといえよう。

他方、生成に関わった者が自らの著作物であるとして主張する可能性も考えられる(67)。そして、このような場合、OSSライセンスの下で広く利用されていたコードに近似するコードが、OSSライセンスの縛りから解放されることによって、逆にproprietaryなコードとして特定の事業者等によって「囲い込まれる」というリスクが生じることも懸念され、このようなリスクの顕在化はOSSコミュニティからすれば極めて不本意な事態であると考えられる。したがって、このような事態が今後頻発するとなれば、次なる対抗策として、OSSライセンスの条件に、学習モデル構築段階での学習データとして利用する場合には、当該学習モデルによって生成されたプログラムコードもOSSライセンスの効果が及ぶことを明示的に追加することも考えられよう。さらに、自らのコードがコード生成AIに利用されることを拒む者が増える場合、今後、OSSライセンスの下での利用条件としてコード生成AIのための学習モデルへの取り込みを禁じるような事項を追加することで、コード生成AIの学習に利用されることに対処することもありうるかもしれない(68)。

6．コード生成AIの拡大から示唆されるプログラム保護の将来的方向性

以上の検討から、アメリカにおけるGitHub Copilotの事例のようなコード生成AIを巡る著作権法を巡る問題のうち、生成段階において、とりわけ検討すべき課題の方向性としては、大まかには2つの系統があると考えられる。

一つは、コンテンツ系と同様、依拠性の評価がどのようになされるべきかという問題である。学習モデルに蓄積された情報をベースとして、入力されたプロンプトに適応した出力データを生成することについて、私見としては依拠性を直ちに肯定することはできないという立場をとるものであるが、そのように解さない

(66) 生成AIによる生成物の表現の著作物性については、平嶋・前掲注2・64－67頁で、大まかではあるが検討した。
(67) いわゆる僭称コンテンツ問題の一類型の発生ともいえるかもしれない。この問題は、もはや著作権法の解釈論ではなく最終的には立法論による解決に負うところが大きいとも考える。
(68) コード生成AI利用の展開が拡がることに伴って、OSSコミュニティとの関係性が変化して、プログラム開発のエコシステムが大きく変容を受ける懸念が呈されていることを考慮すると、オープンソースコミュニティが、何らかの対応を行ってくる可能性も相応にあるように考えられる。
(69) 例えば、文化審議会著作権分科会法制問題小委員会・前掲注52・34頁は、一定の出力制約のための技術的措置が採られているような場合に限って依拠性が否定されるとする立場とみられ、学説上も現状では依拠性を肯定する立場が多いといえよう。私見としては、積極的に特定の著作物に類似する生成物を出力するような生成処理がなされているような場合は格別、そうでない場合にまで安易に依拠性を肯定することは慎重であるべきと考える。

考え方も当然成り立ちうる(69)といえる。もっともこの問題はコード生成系あるいはコンテンツ生成系でも変わらない、生成AIと著作権法を巡る極めて基盤的な問題として認識できるであろう。

また、コード生成AIによる自動コード生成やコード補完等におけるコードの自動生成の結果として、まとまりのあるプログラムコードを生成する場合ではなく、部分的・補完的に生成されるプログラムコードにおいて、既存のプログラムコードの記述に類似するものが一部含まれるような場合についても、一定の範囲では権利制限の対象にあるものとして扱う余地はあるように考える(70)。この点で、元々は検索サービスへの対応を守備範囲の一部とする規定として設けられた法47条の5のうち、とりわけ3号による対応(71)によって、その役割を担わせることが適切であるようにも思われる(72)。

もう一つは、OSSライセンスの下で要求される事項、とりわけ著作権に係る表示やOSSライセンス文書の付加等について、コード生成AIによって生成されたプログラムコードの提供段階に際しても実現させうるのか、あるいは他の代替的手段をもって実現されたものとして評価しうるのかという問題が挙げられる。もちろん、その前提問題として、生成AIによって創り出されたプログラムコードに対して学習に用いられたプログラムコードに係る権利の効力が及ぶという立場を広く肯定しうるのか否かという問いについて明らかにされる必要がある。

以上、検討してきたコード生成AIにおけるプログラムコードを巡る問題の本質とは、実は多くは著作権自体を巡る問題ではないのかもしれない。OSSライセンスの下でのコード自体は、本来はその利用に対して経済的対価を得ること自体を目的としているものではなく、むしろ、コード生成AIの多くがサブスクリプション型に移行することで有料化が進んでいる実態に鑑みて、事実上OSSの下で提供されているプログラムコードの特徴を利用してproprietaryなコードへの取り込みがなされていることこそが問題の根底にあるともいえる。したがって、法制度として、この点は何らかの制度によって補完する必要はあるのかもしれない。しかし、それが現行の著作権法の枠組みで適切に実現しうるといえるのか、かなり疑問であるようにも考える。

プログラム著作物の場合、そもそも機能的要素が極めて強い著作物であって、コードの表現としての鑑賞性や審美性といった要素が問題となるものではなく、機能性・実用性・合理性といった要素に基因した創作的表現であることから、元来、プログラムを著作物として法的に保護することについてのミスマッチが認識されていた(73)ところであるが、生成AIの出現とプログラム開発過程への普及によって、この問題がより一層顕在化してきているようにも考えられる。将来的な方向性の一つとしては、例えば、プログラムについて著作権法の下での保護対象から異なる法制(74)による法的保護へとシフトさせるといった思い切った制度設計上の転換を行うことが、プログラムの開発・利用にとってむしろ資する可能性が生じてくるようにも思い到るところではある。しかし、そのような結論を導くに足りるほどの十分な合理的な根拠を得るまでには、更なる理論研究と実証過程を要するところであろう。

---

(70) 例えば、生成コードが類似する場合の侵害成否の評価の観点から依拠性が必ずしも否定されないような場合であっても、少なくとも、入力に対して一定の出力制約をするための所定の措置が採られているコード生成AIを用いた生成物については、学習に用いられたプログラムコードの記述の一部と類似する記述を生成されたコードが含まれる場合も、その著作物性の評価如何に関わらず、権利制限対象とすることで、コード生成ツール利用の法的な予見可能性を高めることの意義は大きいものと考える。もちろん、そのような類似性のあるコード表現自体が機能的・技術的要請から著作物性がそもそも否定される場合も少なくないかもしれない。

(71) 3号では対象範囲を政令で定めるものとされている。前掲注70との関連では、出力制約のための一定の措置内容等を政令で指定することによって対応可能であると考える。

(72) 政令以外にも、新たな立法論さらには法政策の観点からも検討を要する問題かもしれない。生成されたコードについて、その後の利用に伴う著作権の効力をどのように扱うべきかという問題についても併せて認識できる。

(73) そもそもプログラムコード自体を著作物として著作権法の下での保護の枠組みに取り込むかを巡って、昭和60年法改正で大きな議論が存在していたことがあらためて想起できるところである（立法の経緯等について中山信弘『著作権法（第4版）』135 − 137頁（有斐閣、2023））。

(74) 一例として、無体物の「商品形態」についても保護対象へ取り入れた令和5年法改正後の不正競争防止法2条1項3号も一つのヒントとなるようにも考える。