JAIST Repository
https://dspace.jaist.ac.jp/

Title	自動運転における車両軌道の抽象化および分析手法の提案
Author(s)	咲川, 麻尋
Citation	
Issue Date	2026-03
Type	Thesis or Dissertation
Text version	author
URL	https://hdl.handle.net/10119/20440
Rights	
Description	Supervisor:青木　利晃, 先端科学技術研究科, 修士(情報科学)

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

修士論文

自動運転における車両軌道の抽象化および分析手法の提案

咲川 麻尋

主指導教員 青木 利晃

北陸先端科学技術大学院大学
先端科学技術専攻
（情報科学）

令和８年３月

Abstract

In recent years, research and development of autonomous driving technology has been actively progressing in both the automotive and information technology industries. Not only has the evolution of advanced driver assistance systems (ADAS) been accelerated, but numerous automakers, parts suppliers, and IT companies around the world are participating in the development of fully autonomous driving (Level 4) under certain conditions, and ultimately Level 5 autonomous driving, which will enable operation in any environment. However, in order to implement and popularize autonomous driving technology in society, significant challenges must be overcome alongside technological development. One of these is "safety verification." Because the system takes over all vehicle recognition, judgment, and control functions from the human driver, it is necessary to objectively and quantitatively demonstrate safety levels equal to or even superior to those of humans.

Relying solely on traditional on-road testing is not a realistic approach to evaluating the safety of autonomous driving systems. Until now, automobile safety evaluations have focused on long-distance driving on test courses or public roads. However, statistically proving the safety of an autonomous driving system requires astronomical driving distances. This not only requires enormous time and expense, but also poses a significant risk of accidents during testing. Furthermore, dangerous situations that could lead to personal injury accidents only occur sporadically in everyday driving environments, making it impossible to cover diverse and complex traffic conditions through real-world driving alone. Therefore, verification based solely on real-world driving has physical and statistical limitations.

To overcome these challenges of real-world driving tests and advance development and evaluation efficiently and comprehensively, the "scenario-based approach" is currently mainstream. This method involves predefining various anticipated traffic situations as "scenarios" and recreating these scenarios in a virtual environment to verify the behavior of autonomous driving systems. However, if each company develops systems based solely on its own unique standards, it becomes difficult to objectively evaluate the safety of the technology and make cross-comparisons. To resolve this issue, it is essential to establish a standard evaluation method and common platform that transcends industries and countries. Against this background, the Japan Automobile Manufacturers Association (JAMA) proposed the "JAMA Framework." This framework aims to comprehensively and systematically evaluate the safety of autonomous driving systems by organizing and classifying events important to safety evaluation from an infinite number of real-world traffic environments as "scenarios." Scenarios in the JAMA framework express driving situations such as "cutting in," "departure," "acceler-

ation," and "deceleration," as well as the vehicle's initial position, using natural language, parameters, and diagrams. This representation format has a "high level" of abstraction, making it easy for humans to intuitively understand traffic situations. However, the raw data (vehicle trajectories) obtained from simulations are huge time-series datasets that record extremely detailed physical quantities such as vehicle position (XYZ coordinates), orientation, speed, and acceleration at very short time intervals. While this is essential for detailed analysis of system behavior, its very "low level" of abstraction prevents direct comparison and verification with high-level scenarios defined by humans. In other words, a large "abstraction gap" exists between the descriptions in the JAMA framework, which serve as evaluation standards, and the simulation data that records the behavior of the autonomous driving system being verified. Existing methods have attempted to bridge this gap by using manual review or rule-based judgments with strict thresholds, but these methods have limitations in terms of efficiency and comprehensiveness. This abstraction gap poses a significant obstacle to efficiently and automatically determining whether simulation results meet specific scenario requirements and to discovering undefined, unknown, and dangerous behaviors (new candidate scenarios) from massive driving logs.

In this paper, we aim to bridge this "abstraction gap" by proposing a method for abstracting low-level vehicle trajectory data and enabling the analysis of driving behavior and situation changes. The proposed method converts trajectories, which are continuous numerical data, into meaningful, discrete "states" and represents their temporal transitions as "symbol sequences." Specifically, we implemented and evaluated the following two approaches. The first is the "equidistant grid method," which divides the relative coordinate space centered on the ego-vehicle into equidistant grids and defines states based on which grids other vehicles are located. The second is the "domain decomposition method," which segments the area around the vehicle into semantic regions such as "forward," "lateral," and "rearward," achieving abstraction that more closely matches human perception. These methods enable the seamless integration of massive amounts of data obtained from simulations and real-world driving with a structured scenario set. To verify the effectiveness of the proposed method, we conducted experiments using a pseudo dataset of tens of thousands of records generated by systematically varying parameters, as well as more realistic driving data obtained from the autonomous driving simulator AWSIM. The evaluation metrics used were scenario detection accuracy, data compression rate, and safety assessment soundness. Furthermore, to clarify the position of this research, we conducted comparative experiments with representative existing approaches. The first comparison target was a formal method using high-precision maps (Lanelet2) and signal temporal logic (STL).

This method determines scenarios based on strict logical formulas, but experiments revealed that it is vulnerable to errors in map definitions (e.g., lane width settings), and even slight differences in settings from the real environment can result in false negatives. The second comparison target was a method using large-scale language models (LLMs), which have attracted attention in recent years. While LLMs excel at understanding the context of trajectory data, they face challenges in applying them to rigorous safety verification due to the computational cost of processing thousands of records and the problem of hallucination, which outputs nonexistent facts.

Experiments showed that the proposed method achieved a high compression rate, reducing the original data volume by approximately 99%. Furthermore, by appropriately setting the grid width and region definition, we were able to reduce false negatives in scenario detection to zero, demonstrating sound abstraction that prevents "overlooking danger," which is paramount in safety verification. In particular, when using the region decomposition method (15-region model), we confirmed that the number of states can be reduced while maintaining the number of false positives compared to the uniform grid method. Furthermore, while the existing method (STL) is a discriminative approach that determines "True/False," our proposed method has the distinct advantage of describing trajectories as symbolic sequences. This makes it possible to analyze and search for transition patterns after the fact, even for unknown behaviors that have not been defined in advance.

In conclusion, our method maintains high robustness while reducing its dependence on high-precision maps, and is capable of faster and more reliable processing than LLM. This makes it extremely effective as a technology for quickly screening dangerous scenarios from massive driving logs, significantly contributing to ensuring the safety of autonomous driving technology and improving development efficiency.

目 次

第 1 章　はじめに **1**

第 2 章　関連研究 **3**
2.1　シナリオベースアプローチ 3
2.2　知識駆動アプローチ 4
2.3　データ駆動アプローチ 5
2.4　軌道の抽象化 6
2.5　本研究の立ち位置と新規性 6

第 3 章　準備 **8**
3.1　抽象解釈 8
3.1.1　データマッピング 9
3.2　JAMA フレームワーク 11
3.3　自動運転システム 12

第 4 章　データマッピングを用いた車両軌道の抽象化 **15**
4.1　具体空間の定義 15
4.1.1　具体空間の軌道 15
4.1.2　シナリオの定義 16
4.1.3　安全性の定義 17
4.2　等間隔のデータマッピング 18
4.2.1　抽象空間の軌道 18
4.2.2　シナリオの定義 20
4.2.3　安全性の定義 21
4.3　自車周囲の車両位置をもとにしたデータマッピング 22
4.3.1　抽象空間の軌道 24
4.3.2　抽象空間における事象検出の正当性 27
4.3.3　安全性の定義 28
4.3.4　安全性検証の健全性 28
4.3.5　15 領域の場合 29

第 5 章　提案手法に基づくツール実装と実験　34
5.1　システム構成 34
5.2　座標系の正規化 35
5.3　時系列データの圧縮アルゴリズム 35
5.4　シナリオ検出ロジック 36
5.5　入力と出力に関して 37

第 6 章　実験　38
6.1　等間隔のグリッドを用いたデータマッピングの実験 38
6.1.1　データセット 38
6.1.2　実験方法 40
6.1.3　実験結果 41
6.2　車両周辺領域を分割したデータマッピング実験 48
6.2.1　実験方法 48
6.2.2　実験結果 49
6.3　実際のシミュレータ軌道での実験 51
6.4　Lanelet2 と STL を用いた手法との比較実験 55
6.5　LLM との比較 61

第 7 章　考察　65
7.1　等間隔のグリッドを用いたデータマッピングの実験結果の考察 . . . 65
7.1.1　シナリオの識別精度 65
7.1.2　シナリオの状態数と圧縮率 66
7.2　車両周辺の領域を分割した手法の実験結果の考察 66
7.2.1　シナリオの識別精度 66
7.2.2　シナリオの状態数と圧縮率 67
7.2.3　9 領域から 15 領域に拡張した場合の効果 67
7.2.4　グリッド手法との比較 68
7.3　安全性分析における評価 68
7.4　提案手法の利点と限界点の考察 69
7.4.1　健全性が保証されていることの利点 69
7.4.2　横方向運動と縦方向運動の抽象化特性 70
7.4.3　複合的なシナリオに対する検出特性の課題 70
7.5　既存手法との比較 71
7.5.1　lanelet2 と STL を用いた手法との比較 71
7.5.2　LLM を用いた手法との比較 72
7.5.3　提案手法と既存手法の特性比較 73

第 8 章　おわりに　76

図 目 次

3.1 kripke 構造を用いた交通信号の抽象化の例 10
3.2 データマッピングの概念図 11
3.3 JAMA フレームワークにおける交通外乱シナリオの表 12
3.4 Autoware のアーキテクチャ 13
3.5 自動運転シミュレータから取得される軌道の例 14

4.1 車両軌跡のグリッド変換の概要 15
4.2 可読性の高い抽象化結果 22
4.3 可読性の低い抽象化結果（抜粋） 22
4.4 JAMA フレームワークにおける周囲の車両位置 23
4.5 9 領域の分割 24
4.6 15 領域の分割 30

5.1 分析ツールのイメージ図 34
5.2 抽象化ツールから出力される車両軌道の例 37

6.1 減速とともにカットインする軌道 54
6.2 減速した後にカットインする軌道 55
6.3 軌道データに対し自由記述での状況説明を求めた場合の LLM 出力 . 62
6.4 車両軌道がカットインしているかを LLM に聞いた場合 63
6.5 提案手法により同一シナリオを抽象化した結果 63
6.6 大量のファイルを処理させた際に発生したエラーの例 64

7.1 蛇行シナリオの抽象軌道の例 70
7.2 U ターンシナリオの抽象軌道の例 71

表 目 次

6.1 実験に使用したシミュレーションパラメータ 40
6.2 シナリオデータ数 40
6.3 実験で用いたグリッド幅の組み合わせ 41
6.4 分析ツールで使用した車両パラメータ 41
6.5 シナリオ判定の混同行列（カットイン） 42
6.6 等間隔のグリッドをを用いたカットインシナリオ分析結果 42
6.7 等間隔のグリッドを用いたカットアウトシナリオ分析結果 43
6.8 等間隔のグリッドを用いた加速シナリオ分析結果 44
6.9 等間隔のグリッドを用いた減速シナリオ分析結果 45
6.10 等間隔のグリッドを用いた状態数と圧縮率（カットイン） 46
6.11 等間隔のグリッドを用いた状態数と圧縮率（カットアウト） 46
6.12 等間隔のグリッドを用いた状態数と圧縮率（加速） 46
6.13 等間隔のグリッドを用いた状態数と圧縮率（減速） 47
6.14 安全性判定の混同行列 47
6.15 各シナリオにおける安全性分析結果（全グリッドサイズ共通） . . . 48
6.16 9 領域の分析結果 49
6.17 15 領域の分析結果 50
6.18 9 領域の状態数と圧縮率 50
6.19 15 領域の状態数と圧縮率 50
6.20 9 領域での安全性評価 51
6.21 15 領域での安全性評価 51
6.22 シミュレーションデータによる各シナリオの混同行列 53
6.23 シミュレーションデータによるシナリオ別の平均状態数と圧縮率 . . 54
6.24 Lanelet2 と STL によるカットインシナリオ分析結果 57
6.25 Lanelet2 と STL によるカットアウトシナリオ分析結果 58
6.26 Lanelet2 と STL による加速シナリオ分析結果 59
6.27 Lanelet2 と STL による減速シナリオ分析結果 60

7.1 提案手法と他手法の特性比較 75

第1章　はじめに

近年，自動車産業および情報技術産業において，自動運転技術の研究開発が盛んに行われている．先進運転支援システム（ADAS）の高度化から，特定の条件下で完全な自動走行を目指すレベル 4，さらにはあらゆる環境での走行を可能にするレベル 5 の実現に向けて，国内外の多くの自動車メーカー，部品サプライヤーに加え，IT 企業も開発に参入している．

この自動運転技術を社会実装し，広く普及させるためには，大きな課題がある．そのうちの一つが「安全性の保証」である．自動運転システムが社会に受け入れられるためには，人間のドライバーに代わってシステムが車両の認知・判断・操作を担う以上，そのシステムが人間以上に，あるいは少なくとも同等以上に安全であることを客観的に示す必要がある．

従来の ADAS 等の安全性評価ではテストコースや公道での長距離走行が中心的な役割を担ってきた．ところが，自動運転システムに対し統計的に十分な安全性を証明しようとすれば，天文学的な走行距離が必要となる．これには膨大な時間とコストを要するだけでなく，試験中に発生しうる事故のリスクも無視できない．加えて，人身事故に繋がるような危険な状況は日常的な走行環境では偶発的にしか発生せず，多様かつ複雑な交通環境のすべてを実走行だけで網羅することは不可能である．したがって，自動運転システムの安全性評価において，そのようなエビデンスを従来の実走行試験のみによって収集するアプローチは現実的ではない [1]．

これらの課題を克服し，開発と評価を効率的かつ網羅的に進めるためのアプローチとして,「シナリオベースアプローチ」が現在の主流となっている．これは，想定される様々な交通状況を「シナリオ」として事前に定義し，仮想環境等の制御された条件下で再現することで，危険な事象を含む多様なケースに対する自動運転システムの挙動を効率的に評価するものである．このアプローチではシナリオ生成 [2]，シナリオ記述形式 [3]，シナリオ分析 [4],[5] 等様々な側面で研究がされている．

シナリオベースアプローチが普及しつつある一方で，各企業が独自の基準で開発を進める現状では，技術の安全性に関する客観的な評価や相互比較が困難という問題が残る． この課題を解決するには，業界や国を超えた標準的な評価手法と共通基盤の確立が不可欠である．こうした背景のもと，一般社団法人日本自動車工業会（JAMA）は、広範囲かつ無限に存在する現実の交通環境から，安全性評価に重要な事象を「シナリオ」として系統的に整理・分類することで，自動運転

の安全性を包括的かつ体系的に評価することを目指した枠組み「JAMA フレームワーク」を提唱した [6].

この JAMA フレームワークにおいて，シナリオは「Cut-in」「Cut-out」「Acceleration」「Deceleration」といった走行状況の分類や，車両の初期位置等を含み，自然言語やパラメータ，図を用いて定義される．これは，人間が交通状況を直感的に理解しやすく「高レベル」な抽象度を持つ表現形式であると言える．

一方，シミュレーションから取得される生のデータ（車両軌道）は，車両の位置（XYZ 座標），向き，速度，加速度といった内容が極めて細かい時間間隔で記録された膨大な時系列データである．これはシステムの挙動を詳細に分析するためには不可欠だが，非常に「低レベル」な表現であり，そのままでは人間が定義した高レベルなシナリオと直接比較・照合することはできない． つまり，JAMA フレームワークと自動運転システムの振る舞いを記録したシミュレーションデータとの間には,「抽象度のギャップ」が存在する．この抽象度のギャップは，シミュレーション結果が特定のシナリオの要件を満たしているかを効率的に判定したり，膨大な走行ログの中から未知の危険な振る舞いのパターン（新たなシナリオ候補）を発見したりする上で，大きな障壁となっている．

そこで本論文では，この「抽象度のギャップ」を埋めることを目的とし，低レベルな車両軌道データを抽象化し，運転行動や状況変化を分析できる手法を提案する．本手法では抽象解釈の一種であるデータマッピングを用いて，軌道データをシナリオレベルまで抽象化する．まず，等間隔のグリッドを用いたデータマッピングにより軌道を抽象化し，次に，車両周辺の領域を分割したデータマッピングにより軌道を抽象化する．2 つの手法に関して，シナリオ検出精度，状態圧縮率，安全性評価精度の観点で，実験を通じて評価を行う．本研究が提案する手法によって，シミュレーションや実走行で得られた膨大なデータと，体系化されたシナリオ群とを円滑に結びつけることが可能となる．これは，シナリオベースでの網羅的な安全性評価を加速させるだけでなく，シミュレーション結果から新たな知見を得ることを容易にし，自動運転技術全体の安全性と信頼性の向上に寄与するものである．

本論文は 7 つの章で構成される．第 2 章では関連研究および先行研究についてまとめ，本研究の立ち位置および新規性について述べる．第 3 章では手法の基盤となる技術要素について説明する．第 4 章でデータマッピングを用いた抽象の手法について述べる．第 5 章では第 4 章の定義をもとに実装したツールについて述べる．第 6 章ではツールを用いて評価実験を行い，既存手法との比較についても実験を行う．第 7 章ではこれまでの実験結果から提案手法に関する考察を行い，利点と課題についてまとめる．最後に 8 章で総括を行い，今後の展望についてまとめる．

第2章　関連研究

自動運転の安全性を保証するために，様々な観点から研究が行われてきた．本章ではまず，自動運転の安全性を高めるための主要な手法である「シナリオベースアプローチ」の概念とその重要性について述べる．次に，シナリオを構築・生成するための 2 つの主要なアプローチである「知識駆動アプローチ」と「データ駆動アプローチ」について，それぞれの特徴と代表的な研究を紹介する．さらに，具体的な車両軌道をシナリオレベルで扱うための「軌道の抽象化」に関する既存研究を概観する．最後に，これらの先行研究と比較した際の本研究の立ち位置および新規性について論じる．

2.1　シナリオベースアプローチ

ADAS などの運転支援システムなどを含め，従来は実際に走行を行い，走行距離を積み上げることで安全性を統計的に証明する「距離ベースアプローチ」が一般的であった．

しかし，Wachenfeld ら [1] は，このアプローチが車両プロトタイプやテストドライバーの確保，評価期間など多くの観点で莫大な経済的コストを要することを指摘し，自動運転レベルが上がるにつれてそのコストは指数関数的に増大するため現実的ではないと論じている．この距離ベースアプローチの有望な代替案として登場したのが「シナリオベースアプローチ」である．シナリオベースアプローチでは，危険な状況や特徴的な交通状況を「シナリオ」として定義・生成し，それをもとに集中的にテストを行う．これにより，効率的な安全性評価が可能となる．シナリオはシステムのリリース判断にとって極めて重要な要素であるため，体系的にもれなく導き出される必要がある．また，シナリオは開発プロセス全体にわたって追跡可能でなければならない．

シナリオの定義や記述に関する共通基盤として，Geyer ら [7] は自動運転の分析のための基本的なオントロジーを提案しており，これらは多くの後続研究で参照されている．例えば，Ego vehicle（自車）や Scenery（背景）などが何を指すのかを明確化することで，研究成果の相互比較が可能になった．また，シナリオの記述レベルに関しては，Menzel ら [8] が 3 つの抽象化レベルを提唱している．本論文で用いる「シナリオ」という言葉は，主にこのうちの「機能シナリオ（Functional Scenario）」を指す．機能シナリオは自然言語で記述される最も抽象度の高いシ

ナリオであり，人間が直感的に理解できる形式である．これに対し,「論理シナリオ（Logical Scenario）」はパラメータ範囲を定義したものであり,「具体シナリオ（Concrete Scenario）」は各パラメータに具体的な値を指定した実行可能なテストケースに相当する．

シナリオベースアプローチにおいて，これらのシナリオをどのように生成・分析するかについては，主に「知識駆動アプローチ」と「データ駆動アプローチ」の 2 つが主流となっている．

2.2 知識駆動アプローチ

知識駆動アプローチとは，専門知識を活用して，自動運転システムの安全性と機能性を評価するテストシナリオを作成，評価，改良するものである．

開発の初期段階では，シナリオは抽象的な言語で記述される．実際にシミュレーションを行う際にはこれらの抽象的なシナリオを詳細化し，シミュレーション環境で使用データ形式に変換する必要がある．Menzel らの研究 [2] では．キーワードベースでシナリオ記述を自動的に詳細化し，テストケース生成の基礎を提供するアプローチを提示している．キーワードベースの記述はまず，パラメータを用いた表現に変換され，パラメータの選択と組み合わせに関する制約が文書化される．次のステップとしてパラメータ表現がシミュレーション環境で必要なデータ形式に変換される．自動運転の安全を保証するためのテストケースを適切に生成する方法について述べられている．

B. Schütt ら [9] が提案しているグラフィカルフレームワークはシナリオ記述をグラフとしても出力しており，これにより機械学習アルゴリズムを用いて記録されたデータからシナリオの生成や抽出，専門家による作成も行うことができる．また，生成したシナリオをレビューすることもできる．シナリオ記述は様々な抽象レベルをサポートしている．シナリオをグラフィカルに視覚化することで理解しやすさが向上するとされている．シナリオを抽象的な記述だけでなく，グラフィカルな記述が様々なユーザーにとっての使いやすさを提供できることが示唆されている．

L. Huang ら [10] が提案しているオントロジーベースの運転シーンのモデリングでは都市環境の運転シーンをモデル化している．また，自動運転車周囲の 8 つの領域の安全性を，TTC,TIV を用いて評価する．これにより意思決定を改善している．この手法はリアルタイム要件を満たすことが示されており，自動運転アプリケーションの意思決定モジュールとして使われることが期待される．

L. Kovacs ら [11] は，シナリオのカバレッジ測定や合成を行うためには交通シナリオの形式的記述が必要であるとし，グラフ変換規則を用いた数学的に正確な動作形式化を提案している．また，提案する形式化が Scenic と CARLA シミュレータでシミュレートできることを示した．長期目標としては交通状況に対する様々な

カバレッジメトリクスを定義すること，交通要件に対応する完全なテストスイートを体系的に合成することであると述べられている．

D. J. Fremont ら [12] は，シミュレーション評価と実世界でのテストの双方にまたがる，自動運転車の安全性テストへの新たなアプローチを提案している．この手法は形式手法に基づいており，シナリオと安全性特性の形式仕様記述，形式シミュレーションを用いたアルゴリズムによるテストケース生成，実路試験（トラックテスト）用のテストケース選択，実路での実行，そして結果データの分析を統合したものである．産業用テスト施設での実車実験により，形式シミュレーションが実路で実行すべきテストケースの特定に効果的であること，そしてシミュレーションと実世界との間のギャップを体系的に評価し埋めることができるという仮説を裏付けている．

知識駆動アプローチにおけるメリットとしては，オントロジーを使用する場合，オントロジーから得られるシーンを確認する必要はなく，最初のオントロジーとなっている部分のみを確認すればよい点にある．

2.3 データ駆動アプローチ

データ駆動アプローチでは機械学習やパターン認識といった手法が用いられることが多い．時間的離散化や大規模なテストデータを集約・分析するためにクラスタリングが用いられている．これにより関連するシナリオを特定することが可能になる．

L. Klitzke ら [13] の研究では大規模な走行データを管理するシステムを提案している．このシステムを用いることで車線変更シナリオの検出を行っている．このシステムは，シナリオマイニングの基盤となる．

M. Qianli ら [14] の研究ではスライディングウィンドウ処理によって運転データをシナリオサンプルに分割する．次に，LSTM ベースの手法を用いて，速度，加速度，ステアリングホイール角などの多変量時系列から特徴を抽出し，重要なシナリオの認識を容易にする．最後に，K 平均法クラスタリングと人間が注釈を付けたシナリオとの類似性分析を用いて，シナリオを自動分類している．

ほかにも，M. Kang ら [15, 16] の研究では 3D LiDAR 点群データ（3D-LiDAR PCD）を含む実際の走行データを用いて，車両速度，加速度，検出された重大事故車両（ATV）などの運動特性を抽出したり，運動特性の重要性を評価し，シナリオへの設定値の選択基準を設定したりしている．また，自動運転システムが事故に対して脆弱になる重要な瞬間にキャプチャされたセンサーデータから，統計的に有意でより詳細なシナリオを抽出している．

2.4　軌道の抽象化

シナリオベースアプローチを効果的に運用するためには，上記のような「シナリオ生成」だけでなく，生成されたシナリオや走行結果（軌道データ）を効率的に評価・分類するための「軌道の抽象化」技術が不可欠である．具体空間の連続値（座標や速度）のままでは，人間が直感的に理解したり，大量のデータから特定の意味を持つ挙動を検索したりすることが困難だからである．

軌道の抽象化に関する研究として，Reimann ら [17] の研究がある．彼らは信号時相論理（STL）などの形式手法を用いて，具体的な軌道データが特定のシナリオ定義（論理式）を満たすかどうかを判定している．安全性の判定には RSS（Responsibility-Sensitive Safety）距離を使用しており，数学的に厳密な評価が可能である．しかし，この手法は「事前に定義したシナリオ（論理式）に合致するか否か」を判定する（判別的アプローチ）ものであり，定義外の挙動や，未知のシナリオパターンを記述・発見することには適していない．

また，Muslim ら [18] は，実世界のデータセットからカットアウトシナリオを抽出し，車両の運動学的パラメータ（車間距離や速度差など）の分布を定量化している．これはデータからパラメータ範囲を学習する点では有用であるが，時系列的な状態遷移（例えば「徐々に接近してから急に割り込む」といった一連の流れ）としての構造化までは踏み込んでいない．

2.5　本研究の立ち位置と新規性

以上の先行研究を踏まえ，本研究の立ち位置と新規性について述べる．本研究は，シミュレーション等で得られた具体空間の軌道を，人間が解釈可能な形式へ変換するという点で，広い意味での「知識駆動アプローチ」の支援技術に位置づけられる．既存研究との主な相違点および新規性は大きく分けて以下の 2 点である．

第一の新規性は，より，汎用的な抽象化手法を提案している点にある．Reimann ら [17] のような STL を用いた従来手法は，データが特定のシナリオか否かを判定する「判別器」としての役割が主であった．これに対し本研究では，すべての軌道データを「状態遷移のシンボル列」として抽象化し，記述するアプローチを採用している．この手法により，事前にシナリオを厳密に定義せずとも，軌道データ自体から「どのような挙動が行われたか」を事後的に読み解くことが可能となる．これは，予期せぬ挙動の発見や，大量データからの類似シナリオ検索において大きな利点となる．

第二の新規性は，グリッドおよび領域分割を用いた直感的な可視化・分析基盤を提供している点である．Muslim ら [18] のような単なるパラメータ抽出にとどまらず，本研究では「空間的な位置関係の遷移」を明示的に扱う．これにより，JAMA フレームワークが定義するような「カットイン」や「カットアウト」といった高レ

ベルな機能シナリオと，低レベルな軌道データとの間に存在する「抽象度のギャップ」を直接的に埋めることが可能となる．

第3章　準備

第 2 章では関連研究について述べた．本章では本研究で用いる技術的要素である抽象解釈，データマッピング，JAMA フレームワーク，自動運転システムについて説明することを目的とする．

3.1　抽象解釈

抽象解釈理論は，1977 年に P. Cousot と R. Cousot によって提唱され [19]，プログラムの静的解析分野において，広く用いられている．抽象解釈は，コンピュータサイエンスにおいて様々な用途に広く用いられている強力かつ汎用的な理論である．抽象解釈手法が応用されている主要な分野を以下に示す．

抽象解釈の最も主要な応用分野は，プログラムの静的解析である．不変条件（invariants）のようなプログラムの実行時特性を自動的に推論することで，プログラムの正当性や安全性の検証に広く用いられてきた [20, 21, 22, 23]．特に，プログラムの実行可能な状態空間を解析し，リアルタイムプログラムの最悪実行時間（WCET）の境界を決定する研究 [22] や，安全性（Safety）と活性（Liveness）の両方の性質を検証する研究 [20] などでその有効性が示されている．また，得られた解析結果は，コンパイラによる最適化やコードの並列化といったプログラム変換を支援するためにも活用される．

抽象解釈による解析の精度や対象とする特性は，用いる「抽象領域（Abstract Domain）」によって決定される．Chang ら [24] は，特定の解析目的に特化し，計算コストと精度のバランスをとった様々な抽象領域を提案してきた．例えば，オブジェクト指向プログラムの解析を強化するために，項上のスキーマに拡張する合同閉包抽象領域や，与えられた抽象領域からプログラムのヒープ内の特性を推論することを可能にするヒープ継承抽象領域などを提案している．

また，単一の抽象領域では十分な精度が得られない場合，抽象解釈を制約プログラミング（Constraint Programming）のような他の手法と組み合わせるハイブリッドアプローチも存在する [25]．このアプローチは，異なる手法の長所を組み合わせることで，解析の近似を大幅に洗練させ，静的解析の精度を向上させることを可能にする．

現在も抽象解釈の理論と応用に関する研究は活発でありプログラム特殊化手法をより洗練された抽象領域へ拡張することによる最適化など，に使われている．抽

象解釈は，複雑なシステムの挙動を近似・検証するための不可欠なツールであるといえる．

3.1.1 データマッピング

データマッピングとは，プログラム変数や物理量が取りうる値の集合（具体領域）を，より小さなサイズの集合（抽象領域）へと写像することで，状態空間を削減する手法である [26]． これは抽象解釈の一種であり，各変数に対して適切な抽象化関数を定義し，変数の具体的な値を抽象的な表現（抽象値）に変換することで，プログラムやシステムの挙動を高レベルで解析することを可能にする．

ある変数 x が取りうる具体的な値の集合を D（具体領域），抽象化された値の集合を $\hat{D}$（抽象領域）とする．このとき，データマッピングは具体値 $v \in D$ を抽象値 $\hat{v} \in \hat{D}$ へと変換する抽象化関数 $\alpha : D \to \hat{D}$ によって定義される．

通常，シミュレーション等で扱われる具体領域 D（例：実数 $\mathbb{R}$ や浮動小数点数）の要素数は無限であるため，そのまま全状態の組み合わせを探索すると状態爆発が発生する．しかし，要素数が有限かつ小規模な $\hat{D}$ へマッピングを行うことで，解析対象となる状態空間を劇的に削減し，大規模なシステムの検証を現実的な計算コストで実行できる．

Kripke 構造　例として信号機をモデル化すると，Kripke 構造間のシミュレーション関係として形式的に定義できる．システムの振る舞いを記述するモデルとして，以下の 4 組で構成される Kripke 構造 M を定義する．

定義 3.1 (Kripke 構造の定義)**.**

$$M = (S, S_0, R, L)$$

ここで，各要素は以下の通りである．

- S: 状態の有限集合（または無限集合）
- $S_0 \subseteq S$: 初期状態の集合
- $R \subseteq S \times S$: 遷移関係．$(s, s') \in R$ は状態 s から s' への遷移が可能であることを示す．
- $L : S \to 2^{AP}$: ラベル付け関数．各状態 s において成立する原子命題（Atomic Proposition）の集合を割り当てる（例：「青色である」など）．

具体モデルと抽象モデル　例では，信号機の物理的な挙動を具体モデル M_C，信号機による車両のとるべき挙動を抽象モデル M_A とする．

具体モデル $M_C = (S_C, S_{C0}, R_C, L_C)$　S_C は物理状態 {green,yellow,red} の集合である．R_C は微小時間に基づくステップごとの遷移を表す.

抽象モデル $M_A = (S_A, S_{A0}, R_A, L_A)$　S_A は提案手法によって離散化されたシンボル {GO, STOP} の集合である.

データマッピング（抽象化関数 $\alpha : S_C \to S_A$）によって，具体状態 $s_c \in S_C$ は抽象状態 $s_a = \alpha(s_c)$ に写像される．このとき，本手法の抽象化が健全であるためには，すなわち，具体モデル上の任意の遷移 $(s_c, s'_c) \in R_C$ に対して，対応する抽象モデル上の遷移 $(\alpha(s_c), \alpha(s'_c)) \in R_A$ が必ず存在しなければならない.

$$\forall s_c, s'_c \in S_C, \quad (s_c, s'_c) \in R_C \implies (\alpha(s_c), \alpha(s'_c)) \in R_A \tag{3.1}$$

この条件が満たされるとき，具体モデル M_C 上のあらゆる実行パスは，抽象モデル M_A 上のパスとして包含される（過近似される）.

さらに，抽象化関数 α は，対象とする特性（安全性プロパティなど）を保存するように設計される必要がある．具体的には，着目する特性関数 f に対し，以下の条件を満たすことが求められる.

$$c, c' \in D \text{ に対し}, \quad \alpha(c) = \alpha(c') \implies f(c) = f(c') \tag{3.2}$$

すなわち，同じ抽象状態にマッピングされる具体状態は，検証したい特性に関しても等価でなければならない．図 3.1 に kripke 構造を用いた交通信号の抽象化したモデルを示す.

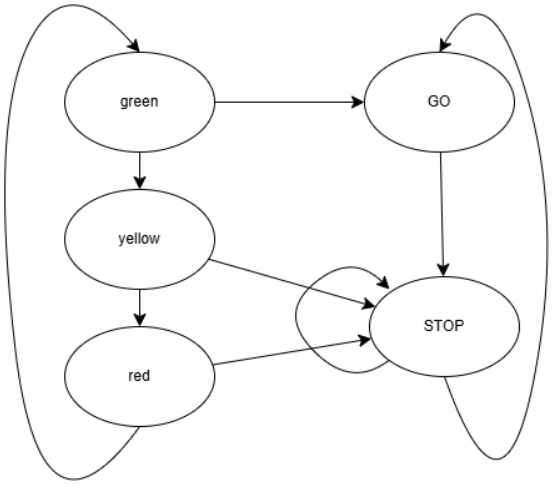


図 3.1: kripke 構造を用いた交通信号の抽象化の例

データマッピングの具体例　例えば，車両の座標などの数値変数を扱う際，厳密な浮動小数点数ではなく，区間などの抽象領域にマッピングすることで，演算誤差を許容しつつ衝突の有無といった安全性プロパティを検証することが可能になる.

代表的なデータマッピングの例として，整数型の変数 v を $\{\mathrm{NEG}, \mathrm{ZERO}, \mathrm{POS}\}$ の3つの状態に分割する符号抽象化（Sign Abstraction）がある．

$$\alpha_{\mathrm{sign}}(v) = \begin{cases} \mathrm{POS}, & \text{if } v > 0 \\ aczero, & \text{if } v = 0 \\ acneg, & \text{if } v < 0 \end{cases} \tag{3.3}$$

このような抽象化を用いることで，プログラムの状態空間を効果的に削減し，解析の計算コストを現実的な範囲に低減することができる（図3.2）．

ただし，抽象化の粒度を粗くすると，実際には安全であるにもかかわらず危険と判定される「False Positive (FP) 」が増加する． このようなトレードオフが存在するため，検証目的に応じて適切な粒度の抽象化関数を設計する必要がある．

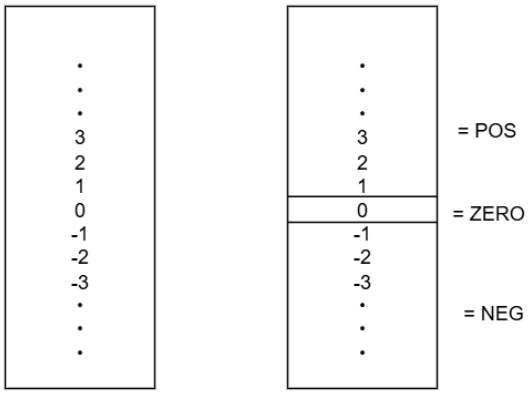


図 3.2: データマッピングの概念図

3.2 JAMA フレームワーク

日本自動車工業会 (JAMA) が発表している JAMA フレームワークは，自動運転システムの安全性を評価するための体系的な枠組みであり，シナリオベースのアプローチを採用している点が特徴である．現実世界の交通環境は極めて複雑であり，すべての状況を実路走行でテストすることは不可能である．そこで，このフレームワークでは，自動運転システムが直面する可能性のある様々な交通状況を「シナリオ」として整理・分類し，仮想環境（シミュレータ）等を用いて効率的に評価を行うことを目指している．しかし，交通環境を分類して組み合わせるだけではシナリオ数が膨大になるため，分析ができない．

そこで JAMA フレームワークでは自動運転の動的運転タスクを実行するために必要なプロセスを物理減速の異なる認知，判断，操作の3要素「認知：認識外乱，判断：交通外乱，操作：車両運動外乱」に分解し，プロセスごとに処理結果に及ぼす要因（root cause）をシナリオ体系として，構造化することで，有限活安全の観点で網羅的な範囲の特定を可能にする．

本研究ではその中でも判断プロセスである交通外乱に着目して研究を行う．交通外乱とは，自車の安全走行が妨げられるような自車周辺に存在する交通参加者の位置と動作である．

JAMA フレームワークにおけるシナリオは，車両の初期位置，速度，加速度，周囲の交通状況（NPC 車両の配置や挙動）といった具体的なパラメータに基づいて定義される．これにより，高速道路での合流や交差点での右折といった特定のユースケースにおいて，自動運転システムが安全マージンを確保できているか，あるいは事故を回避できるかを詳細に評価できる．

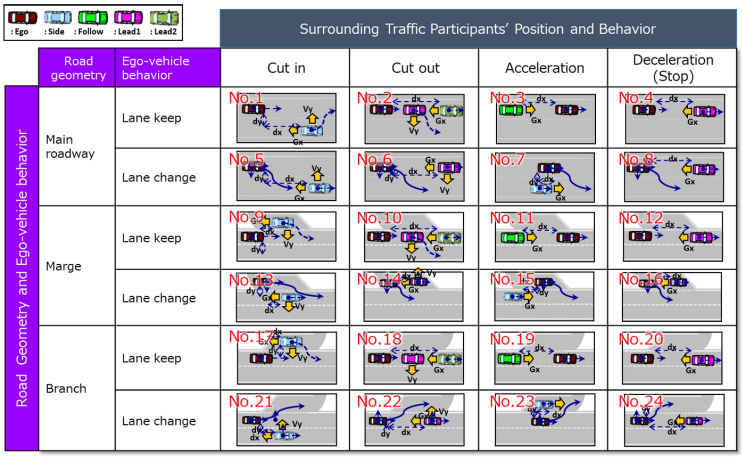


図 3.3: JAMA フレームワークにおける交通外乱シナリオの表 ([6] より引用)

3.3 自動運転システム

自動運転システムとは，車両が人間のドライバーに代わって認知，判断，操作を行うシステムである．車両に搭載されたカメラや LiDAR などを用いて車両周囲の環境を監視し，歩行者，道路上の白線，標識等を認識する．そして，それらの情報に基づきシステムが総合的に判断を下し，アクセル，ブレーキ，ステアリングなどの操作を行う．自動運転のレベル定義としては，運転タスクの主体や走行領域に応じてレベル 0 から 5 までにレベル分けされている米国自動車技術会 (Society of Automotive Engineers) のものがメジャーで，日本でもこれが採用されている．レベル 3 以上では運転の主体がシステムへと移行する．本研究では，システムが主体となって操作を行うレベル 3 以上の自動運転を研究の対象とする．

Autoware　Autoware とはティアフォーが公開している自動運転のオープンソースソフトウェアである [27]．LiDAR，カメラ，GNSS などのセンサーデータを用いた自己位置推定，物体認識，経路生成，車両制御といった自動運転に必要な機能

がモジュール化されて実装されている．実証実験車両や自動運転バスなど，国内外で多くの適用事例がある．Autoware のアーキテクチャを図 3.4 に示す．

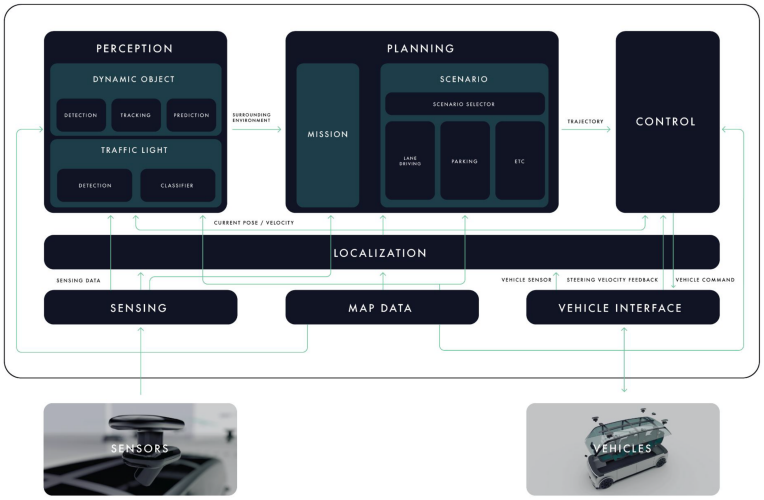


図 3.4: Autoware のアーキテクチャ([28] より引用)

AWSIM　AWSIM は，Autoware の動作検証を行うために開発された，Unity ベースの自動運転シミュレータである．AWSIM に関してもオープンソースソフトウェアとして公開されている [29]．Autoware にもプランニングシミュレータとリプレイシミュレータという 2 つのシミュレーション方法を持っているが，自動運転システム全体をテストすることができないため最終的には実車テストを行う必要がある．AWSIM では LiDAR により仮想空間をとらえることができ，実際の環境に近いシミュレーションが可能となる．

シミュレータから取得される車両軌道　本研究では，上述の AWSIM シミュレータから出力される車両の走行データを分析することを目標とする．取得される軌道データは，図 3.5 に示すように，各時刻における車両の位置（XYZ 座標），向き（オリエンテーション），速度，加速度といった物理量が，極めて細かい時間間隔で記録された時系列データである．これらのデータは，システムの挙動を詳細に解析するためには不可欠な情報源である．しかし，これらは単なる数値の羅列という「低レベル」な表現であり，人間が直感的に理解する「割り込み」や「急停止」といった「高レベル」なシナリオ定義と直接比較・照合することは困難である．そのため，次章で述べる抽象化の手法が必要となる．

```
{
    "ego_estimated_kinematic": [
        {
            "timestamp": 69.883,
            "pose": {
                "position": {
                    "x": 81640.026,
                    "y": 50554.588,
                    "z": 39.246
                },
                "rotation": {
                    "x": -0.375,
                    "y": -0.231,
                    "z": -79.603
                }
            },
            "twist": {
                "linear": {
                    "x": 0.0,
                    "y": 0.0,
                    "z": 0.0
                },
                "angular": {
                    "x": 0.0,
                    "y": 0.0,
                    "z": 0.0
                }
            },
            "acceleration": {
                "linear": {
                    "x": 0.0,
                    "y": 0.0,
                    "z": 0.0
                },
                "angular": {
                    "x": 0.0,
                    "y": 0.0,
                    "z": 0.0
                }
            }
        },
```


図 3.5: 自動運転シミュレータから取得される軌道の例

第4章　データマッピングを用いた車両軌道の抽象化

本章では本研究で提案する車両軌道の抽象化手法について述べる．本研究ではシミュレーションで得られた具体軌道を抽象化し，JAMA フレームワークで用いられているシナリオと比較可能な形式に変換することを目的とする．具体的には，データマッピングを用いて具体軌道を抽象軌道に変換する手法を提案する．

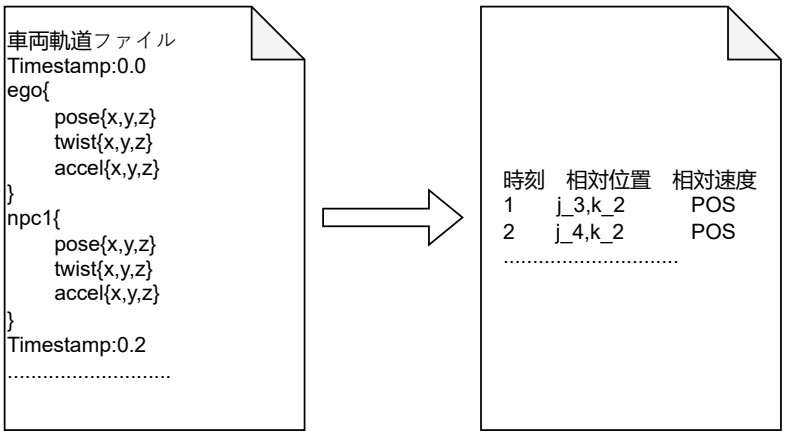


図 4.1: 車両軌跡のグリッド変換の概要

4.1　具体空間の定義

4.1.1　具体空間の軌道

本来の車両軌道には3次元上の座標があり，ego と npc の絶対位置，速度，加速度が記録されている．その中でも，抽象軌道に使用するパラメータを具体空間の軌道として定義する．シミュレータから得られるデータは離散的な時系列データである．本研究では，分析に必要なパラメータを抽出したものを具体空間の軌道として以下のように定義する．

定義 4.1 (具体空間の車両軌道)**.** 具体空間の車両軌道 Trj および，時刻 t_i における車両の状態 s_i を以下の式で定義する．

$$Trj = \{s_0, s_1, s_2, \dots\}$$
$$s_i = (t_i, (x_i, y_i, v_i, a_i)) \in \mathbb{R}_0^+ \times \mathbb{R}^3$$

ここで，軌道の順序は時刻によって定義され，$\forall i, j \in \mathbb{N}, s_i < s_j \iff t_i < t_j$ を満たすものとする．

ここで，t_i はタイムスタンプ，x_i, y_i は自車（ego）に対する他車（npc）の相対位置（x は進行方向，y は横方向），v_i は相対速度，a_i は相対加速度を表す．$y_i > 0$ の場合，npc は ego の左側に存在することを示す．

例 4.1. 例えば，i ステップ目の時刻 $t_i = 3.8$ において，npc が ego の前方 20m，左側 1.5m の位置にあり，相対速度が 0 の場合，状態 s_0 は次のように表される．

$$s_i = (3.8, (20.0, 1.5, 0.0))$$

4.1.2 シナリオの定義

カットイン・カットアウトシナリオ

カットインシナリオおよびカットアウトシナリオは，他車（NPC）が車線境界を跨いで移動する挙動（車線変更）を含むシナリオである．本研究では，自車（Ego）が走行する車線を基準として以下のように定義する．

カットイン NPC が隣接車線から自車線（自車の車幅）へ進入する挙動（前方への割り込み等）．

カットアウト NPC が自車線（自車の車幅）から隣接車線へ逸脱する挙動（前方からの車線離脱等）．

定義 4.2 (具体空間のカットイン)**.** 車長を C_l，車幅を C_w とする．時刻 t_i から t_{i+1} への遷移において，以下の条件 $C_{\text{in}}(i)$ が真となる場合をカットインと定義する．

$$C_{\text{in}}(i) = \begin{cases} \text{true}, & \text{if} \quad (|y_i| \geq \frac{C_w}{2}) \wedge (|y_{i+1}| < \frac{C_w}{2}) \wedge (x_{i+1} \geq \frac{C_l}{2}) \\ \text{false}, & \text{otherwise.} \end{cases} \tag{4.1}$$

カットアウトも同様に以下のように定義される：

定義 4.3 (具体空間のカットアウト)**.**

$$C_{\text{out}}(i) = \begin{cases} \text{true}, & \text{if} \quad (|y_i| < \frac{C_w}{2}) \wedge (|y_{i+1}| \geq \frac{C_w}{2}) \wedge (x_{i+1} \geq \frac{C_l}{2}) \\ \text{false}, & \text{otherwise.} \end{cases} \tag{4.2}$$

定義 4.2, 定義 4.3 は，車両の横位置 y が車幅の半分（車線境界とみなす）の外側から内側へ移動し，かつ縦位置 x が自車よりも前方にある場合を検出するものである．

例 4.2. 車幅 $C_w = 1.9$ の場合，境界値は 0.95 となる．ある時刻で $y_i = 1.2$（左車線），次の時刻で $y_{i+1} = 0.5$（自車線）となった場合，上記の条件を満たしカットインと判定される．

加速・減速シナリオ

加速シナリオおよび減速シナリオは，他車（NPC）の速度変化によって自車（Ego）との車間距離が短縮し，衝突のリスクが生じる状況として定義される．

加速シナリオ Ego の後方を走行している NPC が加速することで，Ego との距離が縮まる状況（後続車による追い上げ）．

減速シナリオ Ego の前方を走行している NPC が減速することで，Ego との距離が縮まる状況（先行車の急ブレーキ等）．

具体空間において「加速」または「減速」が発生し車間距離が車間距離が近づくことを進行方向の相対位置 x を用いて以下のように定義する．車長を C_l, 車幅を C_w とすると

定義 4.4 (具体空間の加速・減速). 具体空間において，相対位置 x の変化に基づき，加速（接近） acc および減速（接近） dec を以下のように定義する．

$$
\begin{aligned}
acc(i) &\iff (x_i < 0) \land (x_i < x_{i+1}) \\
dec(i) &\iff (x_i \geq 0) \land (x_i > x_{i+1})
\end{aligned}
$$

ここで，acc は「後方車両が自車に接近する（相対位置 x が負から 0 に近づく）」ケースを指し，dec は「前方車両が自車に接近する（相対位置 x が正から 0 に近づく）」ケースを指す．

例 4.3. 自車前方の抽象位置 $j_i = 5$（$j_i \geq 0$）において， $j_{i+1} = 4$（$j_i \geq 0$）定義より減速シナリオ $\hat{dec}(i)$ が真となる．抽象軌道上で位置グリッドが変化することににによって加減速状態を識別できることを意味する．

4.1.3 安全性の定義

具体空間における安全を判断する指標として Time To Collision(TTC) を用いる．TTC とは車間距離 (m) を相対速度 (m/s) で割ったものである．この指標は現在の

相対速度が維持された場合にあと何秒で衝突するかを表す指標であり [30]，安全性の検証にも広く使われている．

具体空間において，車両軌道の状態のすべての時間において，y（横）方向の位置が D_l 以上離れているか TTC が 2.0 以上であるときに安全であるとする：

定義 4.5 (具体空間の安全)**.**

$$\mathrm{Safe}(i) = \forall i((|y_i| > D_l) \vee (\mathrm{TTC}(i) \geq 2.0))$$

D_l は以下の計算式で計算される．l_w はレーンの幅を表す

定義 4.6 (D_l の定義)**.**

$$D_l = l_w/2 - v_{wmax} \times 0.4$$

本研究では統一した値を用いる．$l_w = 3.5\,\mathrm{m}, v_{wmax} = 1.8\,\mathrm{m/s}$ より，計算して $D_l = 2.78$ であるとする．

4.2 等間隔のデータマッピング

まず基本的な等間隔のデータマッピングを用いた抽象化を行う．そのために抽象空間の軌道，シナリオ，安全について定義していく．

4.2.1 抽象空間の軌道

シミュレータから得られる具体空間の軌道は，詳細な連続値の時系列データであり，その膨大な情報量ゆえに，人間が直感的に挙動を理解することや，論理的なシナリオとの照合を行うことは困難である．そこで本節では，これらの連続的な物理量を，意味のある離散的な状態へと変換した「抽象軌道」を導入する．これにより，無限に存在する具体的な軌跡を，有限個のシンボル列として扱い，効率的な分析を可能にする．

位置　ego から見た npc の相対的な位置を返す抽象化関数を考える．区間を表現する抽象的な位置の集合 J, K を導入する．区間の幅を w とし，抽象化関数 $\alpha_{posx} : \mathbb{R} \to J$, $\alpha_{posy} : \mathbb{R} \to K$ を次のように定義する：

定義 4.7 (x, y の位置の抽象化関数)**.**

$$\begin{aligned} \alpha_{posx}(x) = j \quad &\Leftrightarrow \quad x \in [jw, (j+1)w) \\ \alpha_{posy}(y) = k \quad &\Leftrightarrow \quad y \in [kw, (k+1)w) \end{aligned}$$

これらを用いて，相対座標 (x, y) を抽象化された位置のペアに写像する関数 $\alpha_{pos} : \mathbb{R}^2 \to J \times K$ を以下のように構成する．

定義 4.8 (位置の抽象化関数)**.**

$$\alpha_{pos}(x, y) = (\alpha_{posx}(x), \alpha_{posy}(y))$$

相対加速度　ego から見た相対的な加速度を返す関数を考える．加速度の抽象集合を $\hat{Acc}$ とし，抽象化関数 $\alpha_{acc} : \mathbb{R} \to \hat{Acc}$ を以下のように定義する：

定義 4.9 (加速度の抽象化関数)**.**

$$\alpha_{\mathrm{acc}}(a_i) = \begin{cases} \mathrm{POS}, & \text{if} \quad a_i > 0 \\ \mathrm{ZERO}, & \text{if} \quad a_i = 0 \\ \mathrm{NEG}, & \text{if} \quad a_i < 0 \end{cases}$$

相対速度　npc と ego の相対速度 v_i は，それぞれの具体空間における速度ベクトル v_{npc}, v_{ego} を用いて $v_i = v_{npc} - v_{ego}$ と定義される．相対速度の抽象集合を $\hat{Vel} = \{\mathrm{POS}, \mathrm{ZERO}, \mathrm{NEG}\}$ とし，抽象化関数 $\alpha_{vel} : \mathbb{R} \to \hat{Vel}$ を以下のように定義する．v_{th} は微小な加減速をノイズとして扱うための閾値である．本研究では v_{th}=0.1 とする．

定義 4.10 (相対速度の抽象化関数)**.**

$$\alpha_{vel}(v_i) = \begin{cases} \mathrm{POS}, & \text{if} \quad v_i > v_{\mathrm{th}} \\ \mathrm{ZERO}, & \text{if} \quad v_{\mathrm{th}} \leq v_i \leq v_{\mathrm{th}} \\ \mathrm{NEG}, & \text{if} \quad v_i < v_{\mathrm{th}} \end{cases}$$

ここで，POS は npc が遠ざかる（または後方から接近する），NEG は npc が前方から接近する（または後方へ遠ざかる）状態を抽象化している．

車両の状態　抽象空間における車両の状態 $\hat{s}_i$ は，位置と相対速度のペアとして以下のように定義される．

定義 4.11 (抽象空間の車両状態（位置・相対速度）)**.**

$$\hat{s}_i = (\hat{p}_i, \hat{v}_i) \in \hat{Pos} \times \hat{Vel}$$

抽象空間における車両の状態 $\hat{s}_i$ は，位置と加速度のペアとして以下のように定義される．

定義 4.12 (抽象空間の車両状態 (位置・加速度))**.** 抽象空間における状態集合 $\hat{S}$, および車両軌道 $\hat{Trj}$ を以下のように定義する：

$$\hat{s}_i = (\hat{p}_i, \hat{a}_i) \in \hat{Pos} \times \hat{Acc}$$

時間と抽象軌道　具体空間の軌道は，実数時間 $t = 0.0, 0.1, \ldots, n$ における具体状態の系列である．これを各時間ごとに抽象化した系列を $\Sigma = \langle \hat{s}_0, \hat{s}_1, \ldots, \hat{s}_n \rangle$ とする．ここで，状態が変化しない連続したステップを集約し，状態が変化した時点でのみ時間が進行するように時間を抽象化する．具体的には，系列 Σ から隣接する重複要素を取り除いた系列への写像 α_{time} を定義する．

定義 4.13 (抽象軌道)**.** 時間抽象化された軌道 $\hat{Trj} = \alpha_{\text{time}}(\Sigma) = \langle \hat{s}'_0, \hat{s}'_1, \ldots, \hat{s}'_m \rangle$ $(m \leq n)$ は，以下の条件を満たす系列である：

1. 順序の保存: $\hat{Trj}$ の各要素は Σ の要素を出現順に抽出したものである．
2. 状態変化の保証: 任意の k $(0 \leq k < m)$ について，$\hat{s}'_k \neq \hat{s}'_{k+1}$ である．

このとき，系列のインデックス $k \in \{0, 1, \ldots, m\}$ を**抽象時間**と呼ぶ．

例 4.4. 区間幅 $w = 2.0\text{m}$ とする．ある時刻において，npc が ego から見て $x = 20.5\text{m}, y = 3.5\text{m}$, 相対速度 $v = 10\text{m/s}$ の状態にあったとする．位置のインデックスは $j = \lfloor 20.5/2.0 \rfloor = 10$，相対速度は閾値を超えているため POS となる．このとき抽象時間 k における状態 $\hat{s}'_k$ は次のように表される．

$$\hat{s}'_k = (k, (10, 3, \text{POS}))$$

4.2.2　シナリオの定義

抽象空間でのシナリオの定義を以下に示す．

カットイン・カットアウトシナリオ

抽象空間のカットイン $\hat{C_{\text{in}}}$ は以下の条件を満たす遷移として定義される．時刻 t_i から t_{i+1} の時に j, k の値が以下の条件を満たすシーンがある場合をカットインとする．

ここで j は x 方向の相対位置をグリッド上に表現したものであり，k の値は y 方向の相対位置をグリッド上に表したものである．定義において，区間幅は 1.0 以上であるとする：

定義 4.14 (抽象空間のカットイン)**.**

$$\hat{C_{\text{in}}}(i) \iff (j_i \geq 2) \wedge \{(k_i \geq 1) \vee (k_i < -1)\} \wedge (j_{i+1} \geq 3) \wedge (-1 \leq k_{i+1} < 1)$$

カットアウトも同様に以下のように定義される：

定義 4.15 (抽象空間のカットアウト)**.**

$$\hat{C_{\text{out}}}(i) \iff (j_i \geq 3) \wedge (-1 \leq k_i < 1) \wedge (j_{i+1} \geq 2) \wedge \{(k_{i+1} \geq 1) \vee (k_{i+1} < -1)\}$$

加速・減速シナリオ

抽象区間の「加速」「減速」は時刻 t_i から t_{i+1} において，以下の条件を満たす遷移として定義される．

定義 4.16 (抽象空間の加速・減速)**.**

$$\hat{acc}(i) \iff (j_i w_j < 0) \wedge (j_i w_j < j_{i+1} w_j)$$
$$\hat{acc}(i) \iff (j_i w_j \geq 0) \wedge (j_i w_j > j_{i+1} w_j)$$

4.2.3 安全性の定義

相対加速度を用いた安全性の定義

抽象空間において安全であると判断された軌道が，具体空間において必ず安全であるように抽象空間における安全を以下のように定義する：

定義 4.17 (抽象空間の安全)**.**

$$\hat{\mathrm{Safe}}(i) = \forall i((k_i < \lceil D_l \rceil \vee \lceil D_l \rceil \leq k_i)$$
$$\vee (j_i \geq 0 \wedge \hat{a}_i = \mathrm{POS}) \vee (j_i < 0 \wedge \hat{a}_i = \mathrm{NEG}))$$

相対速度を用いた安全性の定義

抽象化した相対速度を用いた抽象空間における安全を以下のように定義する：

定義 4.18 (抽象空間の安全（相対速度）)**.**

$$\hat{\mathrm{Safe}}(i) = \forall i\{(k_i < -\lceil D_l \rceil \vee \lceil D_l \rceil \leq k_i)$$
$$\vee (j_i \geq 0 \wedge \hat{v}_i = \mathrm{POS}) \vee (j_i < 0 \wedge \hat{v}_i = \mathrm{NEG})\}$$

抽象軌道の例　以上の定義を用いて得られた抽象軌道について例を用いて説明する．図 4.2 における「抽象空間」列の記述は，グリッドのインデックスを表す記号列である．ここで，i は進行方向（x 軸）のグリッドインデックス (定義における j に相当)，k は横方向（y 軸）のグリッドインデックス（k）を示す．例えば $i50_k3$ という表記は，対象車両が x 方向の 50 番目のグリッド，かつ y 方向の 3 番目のグリッドに位置している状態であることを意味する．グリッドの幅が $1.0, \mathrm{m}$ の場合，具体空間では x 方向に $50.0, \mathrm{m}$ 以上 $51.0, \mathrm{m}$ 未満，x 方向に $3.0, \mathrm{m}$ 以上 $4.0, \mathrm{m}$ 未満の区間の中に npc が存在していることを意味している．

例えば図 4.2 の軌道は 6 つの状態にまで状態数が減少している．このような場合は目で見ても理解しやすい軌道といえる．一方 4.3 のようなデータの場合，元の状

態数よりは半分程度に減少しているが，状態数は 335 となっており，この状態では目で見て分析を行うことが難しい．

Time	抽象空間(Grid)	備考
1(1-16)	i20_k-4	
2(17-25)	i20_k-3	
3(26-32)	i20_k-2	
4(33)	i20_k-1	【カットイン】
5(34-50)	i20_k-1	
6(51-240)	i20_k0	

図 4.2: 可読性の高い抽象化結果

Time	抽象空間(Grid)	備考
1(1-2)	i-50_k3	
2(3-4)	i-49_k3	【加速】
3(5-6)	i-48_k3	【加速】
4(7-8)	i-47_k3	【加速】
5(9)	i-46_k3	【加速】
57(100-101)	i5_k1	
58(102-103)	i6_k1	
59(104)	i7_k1	
60(105)	i7_k0	【カットイン】
61(106-107)	i8_k0	
62(108)	i9_k0	
63(109-110)	i10_k0	
335(599-600)	i282_k0	

図 4.3: 可読性の低い抽象化結果（抜粋）

4.3　自車周囲の車両位置をもとにしたデータマッピング

4.2 節の結果から，状態数を安定的に減らした上で，識別精度を落とさないような抽象化手法について検討する． グリッド手法では等間隔な区間を用いて軌道を表現していたが，本節では JAMA フレームワークとの整合性を高めるため，自車（Ego）を中心とした相対的な位置関係に基づくシナリオ分析を考える．

図 4.4 は，JAMA フレームワークにおいて定義されているシナリオ構造において考慮されるべき自車周囲の車両位置を示している． JAMA フレームワークでは，他車の位置を単なる座標点としてではなく，Lead,Following,Parallel といった相対的な位置として分類している． したがって，抽象化においてもこの分類基準を反映させることで，抽象化される軌道の可読性を高めることができると考えられる．

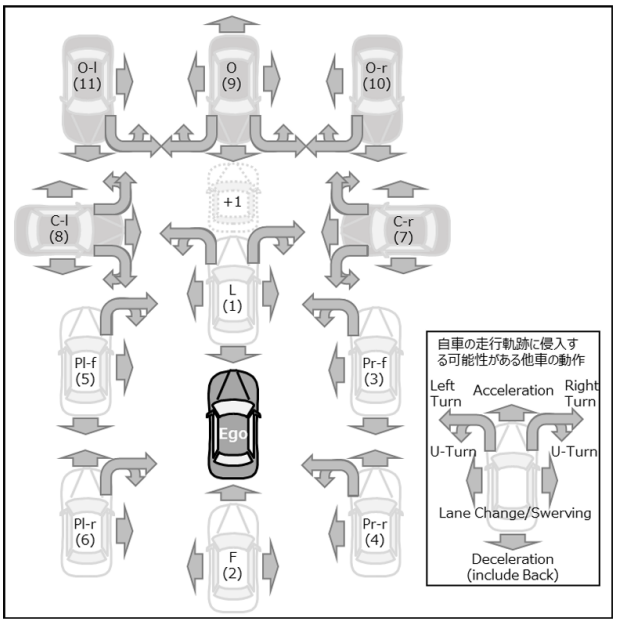


図 4.4: JAMA フレームワークにおける周囲の車両位置 ([6] より引用)

この概念モデルを具体的な空間分割として実装するために，まず図 4.5 のように車両周辺の領域を 9 つに分割した場合を考える．これは，自車位置を中心として，縦方向（前後）を 3 分割，横方向（左右車線）を 3 分割した計 $3 \times 3 = 9$ 領域の構成である．具体的には，自車線上の「前方（Lead）」「自車位置（Ego）」「後方（Follow）」に加え，左右の隣接車線の各領域を定義することで，車線変更や追い越しといった基本的な交通挙動を表現することを意図している．

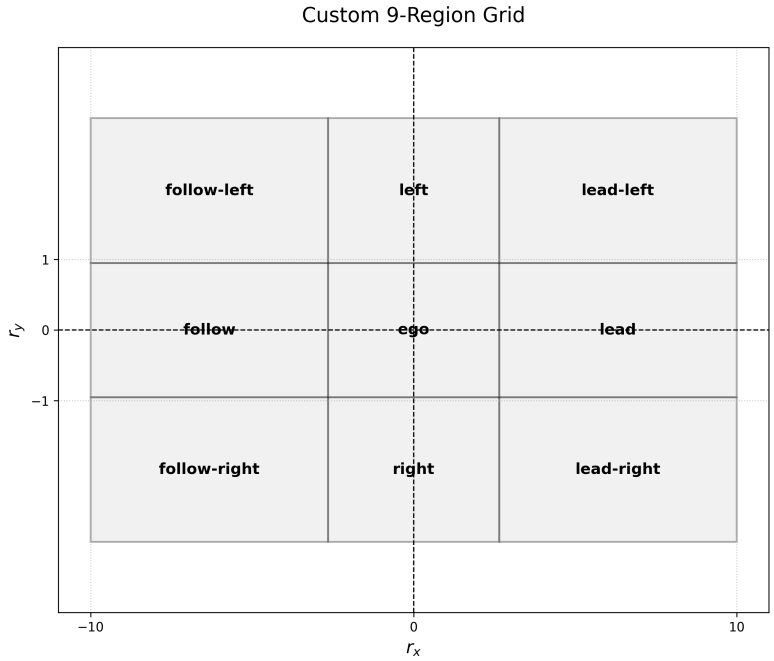


図 4.5: 9 領域の分割

具体軌道の定義は 4.1 節で定義したものを用いる．

4.3.1 抽象空間の軌道

まず 9 個の領域に分割する抽象化を考え，その後 15 個の領域に分割するものを考える．抽象空間の軌道を以下のように定義する．

9 領域の場合 ego から見た npc の相対的な位置を返す抽象化関数を考える．抽象集合 Pos を定義し，$\alpha_{pos} : \mathbb{R}^2 \to Pos$ を次のように定義する．車長を $C_l(\mathrm{m})$，車幅を $C_w(\mathrm{m})$ として ego と npc の相対的な位置を 9 個の領域に分割する:

定義 4.19 (9 領域に分割した位置の抽象化関数)**.**

$$\alpha_{\text{pos}}(x_i, y_i) = \begin{cases} \text{lead-left}, & \text{if} \quad (\frac{C_l}{2} \leq x_i) \wedge (y_i \geq \frac{C_w}{2}), \\ \text{lead}, & \text{if} \quad (\frac{C_l}{2} \leq x_i) \wedge (|y_i| < \frac{C_w}{2}), \\ \text{lead-right}, & \text{if} \quad (\frac{C_l}{2} \leq x_i) \wedge (y_i \leq -\frac{C_w}{2}), \\ \text{left}, & \text{if} \quad (-\frac{C_l}{2} \leq x_i < \frac{C_l}{2}) \wedge (y_i \geq \frac{C_w}{2}), \\ \text{ego}, & \text{if} \quad (-\frac{C_l}{2} \leq x_i < \frac{C_l}{2}) \wedge (|y_i| < \frac{C_w}{2}), \\ \text{right}, & \text{if} \quad (-\frac{C_l}{2} \leq x_i < \frac{C_l}{2}) \wedge (y_i \leq -\frac{C_w}{2}), \\ \text{follow-left}, & \text{if} \quad (x_i < -\frac{C_l}{2}) \wedge (y_i \geq \frac{C_w}{2}), \\ \text{following}, & \text{if} \quad (x_i < -\frac{C_l}{2}) \wedge (|y_i| < \frac{C_w}{2}), \\ \text{follow-right}, & \text{if} \quad (x_i < -\frac{C_l}{2}) \wedge (y_i \leq -\frac{C_w}{2}). \end{cases}$$

定義 4.10 の相対速度を用いる

抽象空間における車両の状態 $\hat{s}_i$ は，位置と相対速度のペアとして以下のように定義される．

定義 4.20 (抽象空間の車両状態（位置・相対速度）)**.**

$$\hat{s}_i = (\hat{p}_i, \hat{v}_i) \in \hat{Pos} \times \hat{Vel}$$

時間　具体軌道において連続する複数の状態が同一の抽象状態に写像される場合，それらを一つの抽象的なステップとして集約する時間抽象化関数 α_{time} を定義する．

各ステップにおける抽象状態を $\hat{s}_i = (\alpha_{pos}(x_i, y_i), \alpha_{acc}(a_i))$ とするとき，

$\alpha_{\text{time}}(\langle \hat{s}_0, \ldots, \hat{s}_n \rangle) = \langle \hat{s}'_0, \ldots, \hat{s}'_m \rangle (m \leq n)$ となる．したがって抽象空間における軌道 $\hat{Trj} = \langle \hat{s}'_0, \ldots, \hat{s}'_m \rangle$ は以下の条件を満たす：

1. 任意の k について，$\hat{s}'_k \in \{\hat{s}_0, \ldots, \hat{s}_n\}$ である．

2. 任意の k について，$\hat{s}'_k \neq \hat{s}'_{k+1}$ である．

ここで，$\hat{Pos}$ は 9 領域に分割された車両位置を表す．

$\hat{S}$ は，この抽象位置と抽象加速度 $\hat{Vel}$ の組み合わせからなる抽象状態全体の集合である．

抽象軌道 $\hat{Trj}$ は，この抽象状態 $\hat{s}'_k$ が時間順序に従って並んだ系列（sequence）として定義される．なお，前述の時間抽象化により，この系列において隣接する要素は必ず異なる状態となる（$\hat{s}'_k \neq \hat{s}'_{k+1}$）．

以上のマッピングにより，抽象空間における軌道 $\hat{Trj}$ は，位置と相対速度の抽象状態の遷移系列として以下のように定義される．

定義 4.21 (抽象空間の車両軌道（9 領域）).

$$\hat{Trj} \subseteq \hat{Pos} \times \hat{Vel}$$
$$\hat{s}'_k = (\hat{p}_k, \hat{v}_k) \in \hat{S}$$
$$\hat{Trj} = \langle \hat{s}'_0, \hat{s}'_1, \ldots, \hat{s}'_m \rangle$$

抽象空間でのカットイン　抽象空間における左車線からのカットイン $\hat{C_{\text{inL}}}$ と右車線からのカットイン $\hat{C_{\text{inR}}}$ を定義する．前方隣接車線（lead-left, lead-right）だけでなく，自車側方の隣接車線（left, right）から前方（lead）への割り込みもカットインとして扱う．

左側からのカットインは，遷移先が lead であり，かつ遷移元が左側の領域である場合として定義される．

右側からのカットインは，遷移先が lead であり，かつ遷移元が右側の領域である場合として定義される．

定義 4.22 (抽象空間のカットイン).

$$\hat{C_{\text{inL}}}(i) = \begin{cases} \text{true}, & \text{if} \quad (\hat{p}_{i+1} = \text{lead}) \wedge (\hat{p}_i \in \{\text{lead-left}, \text{left}\}), \\ \text{false}, & \text{otherwise}. \end{cases}$$

$$\hat{C_{\text{inR}}}(i) = \begin{cases} \text{true}, & \text{if} \quad (\hat{p}_{i+1} = \text{lead}) \wedge (\hat{p}_i \in \{\text{lead-right}, \text{right}\}), \\ \text{false}, & \text{otherwise}. \end{cases}$$

抽象空間でのカットアウト　抽象空間において，左側へのカットアウト $\hat{C_{\text{outL}}}$ は，抽象位置が自車前方領域（lead）から左前方領域（lead-left）へと遷移する場合に発生する．同様に，右側へのカットアウト $\hat{C_{\text{outR}}}$ は，抽象位置が自車前方領域（lead）から右前方領域（lead-right）へと遷移する場合に発生する．

定義 4.23 (抽象空間のカットアウト).

$$\hat{C}_{\text{outL}}(i) = \begin{cases} \text{true}, & \text{if} \quad \hat{p}_i = \text{lead} \ \wedge \ \hat{p}_{i+1} = \text{lead-left}, \\ \text{false}, & \text{otherwise}. \end{cases}$$

$$\hat{C}_{\text{outR}}(i) = \begin{cases} \text{true}, & \text{if} \quad \hat{p}_i = \text{lead} \ \wedge \ \hat{p}_{i+1} = \text{lead-right}, \\ \text{false}, & \text{otherwise}. \end{cases}$$

抽象空間での加速，減速　抽象空間における加速減速 $\hat{acc}, \hat{dec}$ は以下のように定義する

定義 4.24 (抽象空間の加速・減速).

$$\hat{acc}(i) \iff \hat{v}_i = \mathrm{POS} \wedge \hat{p}_i \in \{\text{following}, \text{follow-left}, \text{follow-right}\}$$
$$\hat{dec}(i) \iff \hat{v}_i = \mathrm{NEG} \wedge \hat{p}_i \in \{\text{lead}, \text{lead-left}, \text{lead-right}\}$$

減速シナリオの例について考える

時刻 t_i において，相対速度 $\hat{v}_i = \mathrm{NEG}$ であり，相対位置が lead であったとする．このとき，以下の条件：

$$\hat{v}_i = \mathrm{NEG} \iff \mathrm{NEG} = \mathrm{NEG}$$
$$\hat{p}_i = \text{lead} \iff \text{lead} = \text{lead}$$

が成立するため，定義より $\hat{dec}(i)$ は真となる．

4.3.2　抽象空間における事象検出の正当性

提案手法における抽象空間での事象定義が，具体空間での定義と整合していることを示す．カットインを例として，以下の定理を述べる．

抽象空間において右側へのカットアウト $\hat{C}_{\mathrm{outR}}(i)$ が真であるならば，具体空間においてもカットアウト $C_{\mathrm{out}}(i)$ が真である．すなわち，以下が成立する．

$$\hat{C}_{\mathrm{outR}}(i) \implies C_{\mathrm{outR}}(i)$$

ある時刻 i において，抽象空間で $\hat{C}_{\mathrm{outR}}(i)$ が真であると仮定する．定義より，これは以下の条件と等価である．

$$\hat{p}_i = \{\text{lead-right}\} \quad \wedge \quad \hat{p}_{i+1} = \{\text{lead}\}$$

データマッピング関数 α_{pos} の定義に基づき，この式を具体空間の座標 (x, y) の制約に展開する．まず，$\hat{p}_i \in \{\text{lead-right}\}$ であることから，時刻 i における車両位置 (x_i, y_i) は以下の範囲に存在する．

$$\frac{C_l}{2} \le x_i \le D_{th} \quad \wedge \quad y_i \le -\frac{C_w}{2}$$

$\hat{p}_{i+1} \in \{\text{lead}\}$ であることから，時刻 $i+1$ における車両位置 (x_{i+1}, y_{i+1}) は以下の範囲に存在する．

$$\frac{C_l}{2} \le x_{i+1} \le D_{th} \quad \wedge \quad |y_{i+1}| < \frac{C_w}{2}$$

論理積をとると，以下の条件が得られる．

$$\frac{C_l}{2} \le x_i \le D_{th} \quad \wedge \quad y_i \le -\frac{C_w}{2} \quad \wedge \quad \frac{C_l}{2} \le x_{i+1} \le D_{th} \quad \wedge \quad |y_{i+1}| < \frac{C_w}{2}$$

これは，具体空間におけるカットイン $C_{\mathrm{in}}(i)$ の定義式を包含する．

4.3.3 安全性の定義

具体空間における安全の定義は 4.1.3 節で定義したものを用いる．
抽象空間においては以下のように定義する：

定義 4.25 (抽象空間の安全)**.**

$$\hat{\text{Safe}}(i) \iff \forall i\{(\hat{p_i} \in \{\text{lead}, \text{lead-left}, \text{lead-right}\} \land \hat{v_i} = \text{POS}) \lor (\hat{p_i} \in \{\text{following}, \text{follow-left}, \text{follow-right}\} \land \hat{v_i} = \text{NEG})\}$$

4.3.4 安全性検証の健全性

提案手法による安全性判定が，危険な状態を見逃さない（FN が発生しない）ことを証明する．すなわち，抽象空間で「安全 ($\hat{\text{Safe}}$)」と判定されたならば，具体空間でも「安全 (Safe)」であることを示す.

具体空間における安全性 Safe(i) は,「横方向距離が十分確保されている」または「TTC (Time To Collision) が十分長い」場合に成立すると定義される．

ここで，npc が前方に存在していて，相対速度 v_i が正（$v_i > 0$，すなわち遠ざかる方向）である場合を考える．物理的に，先行車が自車より速く遠ざかっている場合，追突のリスクは無く，TTC は無限大（あるいは定義外の安全状態）となる．したがって，以下の包含関係が成り立つ．C_l を車の長さとすると

$$(v_i > 0 \land x_i > C_l/2) \implies \text{Safe}(i)$$

車が後方にあり，相対速度 v_i が負の場合も同様に考えられる．

$$(v_i < 0 \land x_i < -C_l/2) \implies \text{Safe}(i)$$

安全性 $\hat{\text{Safe}}(i)$ の定義として，以下の条件が含まれている.

$$\hat{p_i} \in \{\text{lead}, \text{lead-left}, \text{lead-right}\} \land \hat{v_i} = \text{POS}$$

ここで，抽象速度 $\hat{v_i} = \text{POS}$ の定義は $v_i > 0$ である．したがって，抽象空間で「前方に車両がおり，かつ遠ざかっている (POS)」と判定された場合，式より具体空間でも必ず安全であることが保証される.

車が後方にいる場合も同様に考えると：

$$\hat{p_i} \in \{\text{following}, \text{follow-left}, \text{follow-right}\} \land \hat{v_i} = \text{NEG}$$

以上のことから，抽象空間での安全条件は，具体空間での安全条件の部分集合（または十分条件）となっている．ゆえに，抽象空間で安全と判定されたシナリオは，具体空間においても安全である．

$$\hat{\mathrm{Safe}}(i) \implies \mathrm{Safe}(i)$$

4.3.5 15 領域の場合

9 領域分割では自車近傍の挙動は捉えられるものの、高速道路等において遠方から接近してくる車両の挙動（危険の予兆）を表現するには粒度が不足する場合がある。 そこで、より広範囲な状況把握を行うため、ego から見た npc の相対的な位置を返す抽象化関数を拡張し、15 領域分割モデルを導入する。

まず、領域の境界を決定するパラメータを定義する。車長を $C_l\,(\mathrm{m})$，車幅を $C_w, (\mathrm{m})$ とし，前方および後方の監視領域（近傍領域）を決定する閾値距離を $D_{th}, (\mathrm{m})$ とする．ここで D_{th} は，想定される最高速度 $v_{ego}, (\mathrm{m/s})$ における車間時間 $t_{th} = 2.0\,(\mathrm{s})$ に基づき，以下の式で設定する．

$$D_{th} = v_{ego} \times t_{th} \tag{4.3}$$

図 4.6 に、これらのパラメータに基づいて定義された 15 領域の分割モデルを示す。本モデルは、9 領域モデルの縦方向（Longitudinal）を拡張し、閾値距離 D_{th} の外側に「遠方領域（Far)」を設けたものである。具体的には、自車位置（Ego）を中心として、縦方向を「遠方前方（Far-Lead)」「前方（Lead)」「自車横（Ego)」「後方（Follow)」「遠方後方（Far-Follow)」の 5 つに分割し、横方向を 3 車線分に分割することで、計 $5 \times 3 = 15$ 領域を構成している。

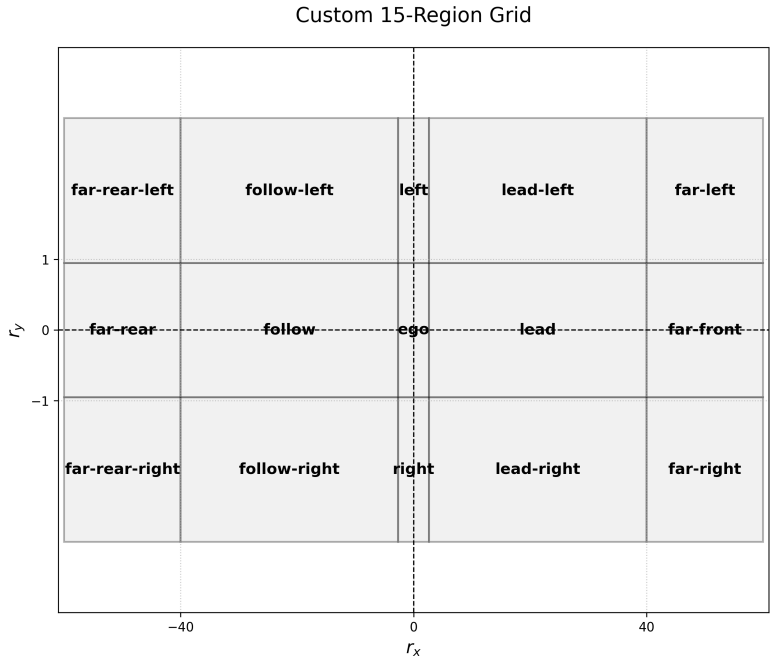


図 4.6: 15 領域の分割

定義したパラメータに基づき，抽象集合 Pos および抽象化関数 $\alpha_{pos} : \mathbb{R}^2 \to Pos$ を以下のように定義する．ここで，(x_i, y_i) は ego から見た npc の相対座標である．

定義 4.26 (15 領域に分割した位置の抽象化関数)**.**

$$\alpha_{\mathrm{pos}}(x_i, y_i) = \begin{cases} \text{far-left}, & \text{if} \quad (x_i \geq D_{th}) \wedge (y_i \geq \frac{C_w}{2}), \\ \text{far-front}, & \text{if} \quad (x_i \geq D_{th}) \wedge (|y_i| < \frac{C_w}{2}), \\ \text{far-right}, & \text{if} \quad (x_i \geq D_{th}) \wedge (y_i \leq -\frac{C_w}{2}), \\ \text{lead-left}, & \text{if} \quad (\frac{C_l}{2} \leq x_i < D_{th}) \wedge (y_i \geq \frac{C_w}{2}), \\ \text{lead}, & \text{if} \quad (\frac{C_l}{2} \leq x_i < D_{th}) \wedge (|y_i| < \frac{C_w}{2}), \\ \text{lead-right}, & \text{if} \quad (\frac{C_l}{2} \leq x_i < D_{th}) \wedge (y_i \leq -\frac{C_w}{2}), \\ \text{left}, & \text{if} \quad (-\frac{C_l}{2} \leq x_i < \frac{C_l}{2}) \wedge (y_i \geq \frac{C_w}{2}), \\ \text{ego}, & \text{if} \quad (-\frac{C_l}{2} \leq x_i < \frac{C_l}{2}) \wedge (|y_i| < \frac{C_w}{2}), \\ \text{right}, & \text{if} \quad (-\frac{C_l}{2} \leq x_i < \frac{C_l}{2}) \wedge (y_i \leq -\frac{C_w}{2}), \\ \text{follow-left}, & \text{if} \quad (-D_{th} \leq x_i < -\frac{C_l}{2}) \wedge (y_i \geq \frac{C_w}{2}), \\ \text{following}, & \text{if} \quad (-D_{th} \leq x_i < -\frac{C_l}{2}) \wedge (|y_i| < \frac{C_w}{2}), \\ \text{follow-right}, & \text{if} \quad (-D_{th} \leq x_i < -\frac{C_l}{2}) \wedge (y_i \leq -\frac{C_w}{2}), \\ \text{far-rear-left}, & \text{if} \quad (x_i < -D_{th}) \wedge (y_i \geq \frac{C_w}{2}), \\ \text{far-rear}, & \text{if} \quad (x_i < -D_{th}) \wedge (|y_i| < \frac{C_w}{2}), \\ \text{far-rear-right}, & \text{if} \quad (x_i < -D_{th}) \wedge (y_i \leq -\frac{C_w}{2}) \end{cases}$$

抽象空間における車両軌道 $\hat{Trj}$ は，位置と相対速度の抽象状態の遷移系列として以下のように定義される．

定義 4.27 (抽象空間の車両軌道（15 領域）)**.**

$$\begin{aligned} \hat{Trj} &\subseteq \hat{Pos} \times \hat{Vel} \\ \hat{s}'_k &= (\hat{p}_k, \hat{v}_k) \in \hat{S} \\ \hat{Trj} &= \langle \hat{s}'_0, \hat{s}'_1, \ldots, \hat{s}'_m \rangle \end{aligned}$$

15 領域におけるシナリオ定義

9 領域の場合と同様に，15 領域に拡張した場合の各シナリオを定義する．

カットイン　抽象空間における左車線からのカットイン $\hat{C_{\mathrm{inL}}}$ および右車線からのカットイン $\hat{C_{\mathrm{inR}}}$ を定義する．前方隣接車線（lead-left, lead-right）や遠方隣接車線（far-left, far-right）だけでなく，自車側方の隣接車線（left, right）から前方領域への割り込みもカットインとして扱う．

左側からのカットインは，遷移先が自車線前方領域（lead または far-front）であり，かつ遷移元が左側の領域（lead-left, far-left, left）である場合として定義される．

右側からのカットインは，遷移先が自車線前方領域であり，かつ遷移元が右側の領域である場合として定義される．

定義 4.28 (15 領域におけるカットイン)**.**

$$\hat{C_{\mathrm{inL}}}(i) = \begin{cases} \text{true}, & \text{if} \quad (\hat{p}_{i+1} \in \{\text{lead}, \text{far-front}\}) \wedge (\hat{p}_i \in \{\text{lead-left}, \text{far-left}, \text{left}\}), \\ \text{false}, & \text{otherwise.} \end{cases}$$

$$\hat{C_{\mathrm{inR}}}(i) = \begin{cases} \text{true}, & \text{if} \quad (\hat{p}_{i+1} \in \{\text{lead}, \text{far-front}\}) \wedge (\hat{p}_i \in \{\text{lead-right}, \text{far-right}, \text{right}\}), \\ \text{false}, & \text{otherwise.} \end{cases}$$

カットアウト　左側へのカットアウト $\hat{C_{\mathrm{outL}}}$ は，自車線前方領域（lead, far-front）から左側領域（lead-left, far-left）への遷移として定義される．

同様に，右側へのカットアウト $\hat{C_{\mathrm{outR}}}$ は，自車線前方領域から右側領域（lead-right, far-right）への遷移として定義される．

定義 4.29 (15 領域におけるカットアウト)**.**

$$\hat{C}_{\mathrm{outL}}(i) = \begin{cases} \text{true}, & \text{if} \quad (\hat{p}_i \in \{\text{lead}, \text{far-front}\}) \ \wedge \ (\hat{p}_{i+1} \in \{\text{lead-left}, \text{far-left}\}), \\ \text{false}, & \text{otherwise.} \end{cases}$$

$$\hat{C}_{\mathrm{outR}}(i) = \begin{cases} \text{true}, & \text{if} \quad (\hat{p}_i \in \{\text{lead}, \text{far-front}\}) \ \wedge \ (\hat{p}_{i+1} \in \{\text{lead-right}, \text{far-right}\}), \\ \text{false}, & \text{otherwise.} \end{cases}$$

加速シナリオ・減速シナリオ　加速・減速シナリオについては，対象となる領域を拡張して定義する．

定義 4.30 (15 領域における加速・減速)**.**

$$\hat{acc}(i) \iff \hat{v}_i = \text{POS} \wedge \hat{p}_i \in \{\text{following}, \text{follow-left}, \text{follow-right}, \text{far-rear}, \text{far-rear-left}, \text{far-rear-right}\}$$

$$\hat{dec}(i) \iff \hat{v}_i = \text{NEG} \wedge \hat{p}_i \in \{\text{lead}, \text{lead-left}, \text{lead-right}, \text{far-front}, \text{far-left}, \text{far-right}\}$$

15 領域における安全性の定義　15 領域を用いた場合の抽象空間における安全性 $\hat{\mathrm{Safe}}(i)$ を以下のように定義する．前方に位置する場合は「遠ざかっている（POS）」こと，後方に位置する場合は「遠ざかっている（NEG）」ことを条件とする．また，遠方領域（Far-front, Far-rear を含む Far 領域全体）においては相対速度に関わらず安全であると仮定する．

定義 4.31 (15 領域における安全)**.**

$$\hat{\text{Safe}}(i) \iff (\hat{p}_i \in P_{\text{front}} \wedge \hat{v}_i = \text{POS}) \vee (\hat{p}_i \in P_{\text{rear}} \wedge \hat{v}_i = \text{NEG}) \vee (\hat{p}_i \in P_{\text{far}})$$

ここで，P_{front}，P_{rear} および P_{far} は以下のような領域集合である**.**

$$\begin{aligned} P_{\text{front}} &= \{\text{lead}, \text{lead-left}, \text{lead-right}\} \\ P_{\text{rear}} &= \{\text{following}, \text{follow-left}, \text{follow-right}\} \\ P_{\text{far}} &= \{\text{far-front}, \text{far-left}, \text{far-right}, \text{far-rear}, \text{far-rear-left}, \text{far-rear-right}\} \end{aligned}$$

第5章　提案手法に基づくツール実装と実験

提案手法に基づいてシナリオを入力とする分析ツールを実装した．定義した抽象化手法および安全性検証を自動化するため，Python を用いた分析ツールを実装した．本節では，そのシステム構成および実装の詳細について述べる．

5.1　システム構成

本ツールの全体的な処理フローを図 5.1 に示す．

1. **データ読み込み・前処理部**: JSON 形式の軌道データを読み込み，座標系の正規化を行う．
2. **抽象化処理部**: 具体空間の連続値（座標，速度）を，第 5.5 節で定義した 9 領域または 15 領域および相対速度ラベルへマッピングする．
3. **時系列圧縮部**: 連続する同一の抽象状態を集約し，イベント駆動型の遷移系列（抽象軌道）へと変換する．
4. **シナリオ・安全性判定部**: 抽象軌道の遷移パターンに基づき，カットイン等のシナリオ発生有無および安全性を判定する．

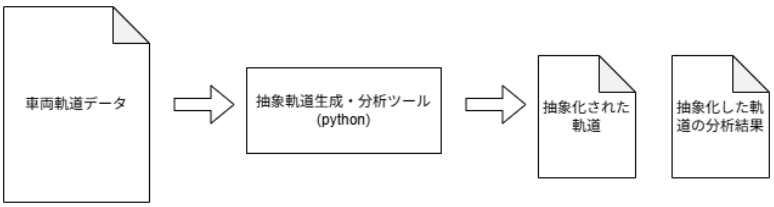


図 5.1: 分析ツールのイメージ図

5.2　座標系の正規化

シミュレータから得られる車両位置 (x, y) は通常，ワールド座標系（地図上の絶対座標）で記述されている．一方，本研究の抽象化手法は自車（Ego）を中心とした相対位置関係に基づく．そのため，実装においては自車の進行方向を基準とした局所座標系への変換処理を行っている．

具体的には，各タイムスタンプ t における自車位置を原点 $(0, 0)$ とし，自車の進行方向ベクトルが x 軸正方向と一致するように，他車（NPC）の座標 (x_{npc}, y_{npc}) および速度ベクトルに対して回転行列 $R(\theta)$ を適用する．

ソースコード 5.1: 座標の正規化と回転処理の実装（抜粋）

```
def normalize_coordinates(ego_coords, npc_coords):

    start_point = ego_coords[0].copy()
    ego_norm = ego_coords - start_point
    npc_norm = npc_coords - start_point


    direction = calculate_direction(ego_norm)
    angle = -atan2(direction[1], direction[0])


    rot_mat = np.array([[cos(angle), -sin(angle)],
                        [sin(angle), cos(angle)]])


    npc_rotated = np.dot(npc_norm, rot_mat.T)
    return ego_rotated, npc_rotated, rot_mat
```

この前処理により，他車が自車の「前方」「後方」「右側」のいずれに存在するかを，幾何学的に正しく判定することが可能となる．

5.3　時系列データの圧縮アルゴリズム

シミュレーションデータは通常 0.1 s などの細かい刻み幅で記録されているが，抽象空間における状態変化はこれに比べて低頻度である．本ツールでは，抽象状態が変化しない区間を一つの「イベント」として集約することで，データの圧縮と可読性の向上を図っている．実装上のロジックとしては，現在の抽象状態 S_t と直前の抽象状態 S_{t-1} を比較し，変化が生じた（$S_t \neq S_{t-1}$）時点でのみデータを記録する手法を用いている．

ソースコード 5.2: 抽象状態の遷移検出と圧縮ロジック

```
for i in range(len(timestamps)):
```

```
2
3     curr_pos, _, _ = get_relative_position(ego[i], npc[i])
4     curr_rv = get_rel_vel_label(rel_vel[i])
5
6
7     if (curr_pos != prev_pos or curr_rv != prev_rv):
8         processed_data.append({
9             'time_range': (start_time, i),
10            'relative_pos': prev_pos,
11            'abstract_rv': prev_rv
12        })
13
14        prev_pos = curr_pos
15        prev_rv = curr_rv
16        start_time = i
```

この処理により，例えば数分間にわたる追従走行（Following）が,「状態：Following」という単一のレコードで表現され，データ量が大幅に削減される．

5.4 シナリオ検出ロジック

生成された抽象軌道データ $\hat{Trj}$ に対し，特定の遷移パターンを検索することでシナリオの検出を行う．例えば，右側車線からのカットインシナリオは，抽象位置が「右前方（lead-right)」または「右遠方（far-right)」から,「前方（lead)」または「自車位置（ego)」へ遷移することとして定義される．

実装では，圧縮された時系列リストを走査し，隣接する状態間での遷移条件を確認することでこれを実現している．

ソースコード 5.3: 抽象遷移によるカットイン検出の実装

```
1 cutin_detected = False
2 for i in range(1, len(processed_data)):
3     prev = processed_data[i-1]['relative_pos']
4     curr = processed_data[i]['relative_pos']
5
6     if prev in ["right","lead-right", "far-right"] and \
7        curr in ["lead", "far-front"]:
8         cutin_detected = True
9         log_scenario("Cut-in␣detected", time=processed_data[i]['time'])
```

このように，抽象化されたシンボル列に対する単純なパターンマッチングによって複雑な交通シナリオを検出できる点が，本手法の実装上の大きな利点である．

5.5　入力と出力に関して

本節では，実装した抽象化ツールの入力および出力データについて述べる．いずれの手法においても，シミュレータから得られた時系列の車両軌道データを主たる入力とし，抽象化されたシンボル列およびその分析結果を出力とする．

グリッドベースの抽象化の場合は入力として，車両軌道，区間幅（x, y）が必要となる．出力としては，抽象軌道（グリッド），分析結果が出力される．車両周辺領域の場合は入力として車両軌道が必要となる，出力としては，抽象軌道（車両周辺領域）と，分析結果が出力される．

図 5.2 に，グリッドベースの抽象化ツールから出力される車両軌道の具体例を示す．ここでは，各タイムステップにおける抽象状態が時系列順に記録され，備考の欄に分析結果が表示されている．

Time	抽象空間(Grid)	備考
1(1-3)	i2_k1	【カットイン】 【減速中】
3(5-8)	i1_k0	【減速中】
4(9)	i0_k0	【減速中】
5(10-13)	i0_k0	【減速中】
6(14-17)	i-1_k0	【減速中】
7(18-21)	i-2_k0	【減速中】
8(22-25)	i-3_k0	【減速中】
9(26-29)	i-4_k0	【減速中】
10(30-33)	i-5_k0	【減速中】
11(34)	i-5_k0	【減速中】
12(35-38)	i-6_k0	【減速中】
13(39-42)	i-7_k0	【減速中】
14(43-46)	i-8_k0	【減速中】
15(47-50)	i-9_k0	【減速中】
16(51-55)	i-10_k0	【減速中】
17(56-59)	i-11_k0	【減速中】
18(60-63)	i-12_k0	【減速中】
19(64-67)	i-13_k0	【減速中】
20(68-71)	i-14_k0	【減速中】
21(72-76)	i-15_k0	【減速中】
22(77-80)	i-16_k0	【減速中】
23(81-84)	i-17_k0	【減速中】
24(85-88)	i-18_k0	【減速中】
25(89-92)	i-19_k0	【減速中】
26(93-97)	i-20_k0	【減速中】
27(98-101)	i-21_k0	【減速中】
28(102-105)	i-22_k0	【減速中】
29(106-109)	i-23_k0	【減速中】
30(110-113)	i-24_k0	【減速中】
31(114-118)	i-25_k0	【減速中】
32(119-121)	i-26_k0	【減速中】

図 5.2: 抽象化ツールから出力される車両軌道の例

第6章　実験

ツールを用いて，シナリオの検出精度，状態数の圧縮率，安全性判定の精度を評価する．既存手法との比較実験についても行う．

6.1　等間隔のグリッドを用いたデータマッピングの実験

目的　第 4.2 節で定義した「等間隔グリッドによる抽象化手法」の基礎特性を明らかにするために評価実験を行う．本実験の主な目的は以下の 3 点である．

1. シナリオ検出精度の評価: 抽象化された軌道データから，元のシナリオ（カットイン等）をどの程度の精度で復元・特定できるかを検証する．
2. データ圧縮率の評価: 元の時系列データと比較して，状態数がどの程度削減されたかを定量的に評価し，可読性とのトレードオフを確認する．
3. 安全性判定の健全性の確認: 危険なシナリオを誤って安全と判定する FN（False Negative）が発生しないかを確認し，安全性検証手法としての健全性を評価する．

6.1.1　データセット

本実験では，シミュレータで取得したデータの代わりに，疑似的に生成したデータセットを使用する．

シミュレーション設定　シミュレーション環境には，自車（Ego）と他車（NPC）の 2 台の車両モデルを配置する．各車両の挙動は，表 6.1 に示すパラメータ範囲内で決定され，軌道が生成される．

収録シナリオの種類　JAMA フレームワーク等の標準的なシナリオ定義に基づき，以下の 4 種類の挙動を含むデータセットを構築した．

1. **Cut-in**: NPC が隣接車線から Ego の前方へ割り込む挙動．
2. **Cut-out**: Ego の前方を走行する NPC が隣接車線へ逸脱する挙動．

3. **Acceleration**: 同一車線内で NPC が加速し，Ego との距離が開く挙動.

4. **Deceleration**: 同一車線内で NPC が減速し，Ego が接近する挙動.

データフォーマット　生成された各軌道データは，ソースコード 6.1 に示す JSON 形式で保存される．各ステップには，時刻（timestamp）に加え，Ego および NPC の絶対座標 (x, y)，速度 (v_x, v_y)，および加速度 (a_x, a_y) が記録されている．また，`groundtruth_vehicles` リストには，シナリオに関与する全ての他車情報が格納されており，ID によって個別に追跡可能な構造となっている．

ソースコード 6.1: "シナリオデータの JSON 形式（構造）"

```
"timestamp": round(t, 2),
    "groundtruth_ego": {
        "pose": {
            "position": {"x": round(ego_x, 3), "y": round(ego_y, 3), "z
                ": 0.0},
            "rotation": {"x": 0.0, "y": 0.0, "z": 0.0}
        },
        "twist": {"linear": {"x": round(ego_v, 3), "y": 0.0, "z":
            0.0}}
    },
    "groundtruth_vehicles": [
        {
            "name": f"npc_{scenario_id}",
            "pose": {
                "position": {"x": round(npc_x, 3), "y": round(npc_y, 3)
                    , "z": 0.0},
                "rotation": {"x": 0.0, "y": 0.0, "z": round(rot_z, 3)}
            },
            "twist": {"linear": {"x": round(npc_v_x, 3), "y": round(
                npc_v_y, 3), "z": 0.0}},
            "accel": {"x": round(npc_a_x, 3), "y": round(npc_a_y, 3), "
                z": 0.0}
        }
    ]
```

上記のデータ構造に基づき，パラメータの値を変化させ，車両軌道データを生成した．シミュレーション環境および車両モデルの固定パラメータとして，シミュレーションの刻み時間 Δt は $0.1\,\mathrm{s}$，対象とする車両サイズは全長 $C_l = 5.3\,\mathrm{m}$，全幅 $C_w = 1.9\,\mathrm{m}$ と設定した．また，各シナリオは自車の縦方向位置が $X_{\mathrm{end}} = 200.0\,\mathrm{m}$ に到達した時点で終了とした．

変動するパラメータとしては，カットイン・カットアウト等を網羅的に評価するために，自車速度 v_{ego} を $10\,\mathrm{km/h}$ から $60\,\mathrm{km/h}$ まで $10\,\mathrm{km/h}$ 刻みで変化させ，他車速度 v_{npc} についても低速から高速域（$30, 50, 70\,\mathrm{km/h}$）を設定した．また，他

車の初期位置 $(x_{\mathrm{npc},0}, y_{\mathrm{npc},0})$ や，車線変更を開始するトリガー位置 x_{trigger}，および車線変更動作に要する時間 t_{lc} を段階的に変化させることで，網羅的な走行パターンの生成を行った．なお，生成において自車と他車の初期位置が物理的に重複する物理的に矛盾するケースはデータセットから除外している．

本実験で使用した詳細なパラメータ設定値を表 6.1 に，生成した各シナリオのデータ数を表 6.2 に示す．

表 6.1: 実験に使用したシミュレーションパラメータ

パラメータ項目	記号	設定値
シミュレーション刻み時間	Δt	$0.1\,\mathrm{s}$
車両サイズ（長 × 幅）	$C_l \times C_w$	$5.3 \times 1.9\,\mathrm{m}$
シミュレーション終了地点	X_{end}	$200.0\,\mathrm{m}$
自車速度 (Ego Speed)	v_{ego}	$10, 20, \ldots, 60\,\mathrm{km/h}$
他車初期位置 (x 座標)	$x_{\mathrm{npc},0}$	$-50, -40, \ldots, 50\,\mathrm{m}$
他車初期位置 (y 座標)	$y_{\mathrm{npc},0}$	$\pm 3.5\,\mathrm{m}$
他車速度 (NPC Speed)	v_{npc}	$30, 50, 70\,\mathrm{km/h}$
割り込み開始トリガー位置	x_{trigger}	$0, 20, \ldots, 100\,\mathrm{m}$
加速開始時間	t_{accel}	$2.0, 4.0\,\mathrm{s}$
減速開始時間	t_{decel}	$2.0, 4.0\,\mathrm{s}$
加速度	a_{npc}	$\pm 2.0, \pm 4.0\,\mathrm{m/s^2}$
目標速度	v_{target}	$30, 50, 70\,\mathrm{km/h}$
車線変更所要時間	t_{lc}	$3.0, 5.0\,\mathrm{s}$

本実験で生成したシナリオデータ数を表 6.2 に示す．

表 6.2: シナリオデータ数

シナリオ種類	データ数
カットインシナリオ	3294
カットアウトシナリオ	4108
加速シナリオ	14952
減速シナリオ	1357

6.1.2 実験方法

節 4.1 で定義した具体空間のデータを，節 4.2 で定義した抽象空間（グリッド）へマッピングし，シナリオ検出の精度および状態数の圧縮率を評価した．

実験に使用したグリッド幅の組み合わせを表 6.3 に示す．ここで，車長 $C_l = 5.3\,\mathrm{m}$ および車幅 $C_w = 1.9\,\mathrm{m}$ を基準とし，車両サイズに対する比率を考慮してグリッドサイズを選定した（表 6.4 参照）．

paragraph 評価指標シナリオ検出精度については，具体空間において定義したラベルと，抽象化データから判定されたシナリオの一致率を混同行列（Confusion Matrix）を用いて評価する．特に，安全性検証においては危険なシナリオを見逃すことが致命的であるため，False Negative（FN：危険なシナリオを安全と誤判定）の発生有無を重視する．

また，データ圧縮率 R_{comp} は，元の時系列データのステップ数 N_{raw} と，抽象化によって集約された状態遷移列の長さ $N_{abstract}$ を用いて，次式で定義する．

$$R_{comp} = \left(1 - \frac{N_{abstract}}{N_{raw}}\right) \times 100\,[\%] \tag{6.1}$$

この値が高いほど，データ量が削減され，検証の効率化に寄与することを示す．

表 6.3: 実験で用いたグリッド幅の組み合わせ

Grid x	Grid y
1.00	1.00
1.75	1.75
2.00	2.00
2.65	0.95
3.50	3.50
4.00	4.00
5.30	1.90

表 6.4: 分析ツールで使用した車両パラメータ

パラメータ項目	記号	設定値
車長	C_l	5.3 m
車幅	C_w	1.9 m

6.1.3 実験結果

シナリオ検出の精度および状態数の圧縮に関する実験結果　各グリッドサイズにおけるシナリオ判定結果の混同行列を表 6.5 以降に示す．表中の TP (True Positive) は正しくシナリオを検出できた件数，FN (False Negative) はシナリオを見逃した件数，FP (False Positive) は無関係な挙動を当該シナリオと誤検出した件数，TN (True Negative) は正しく除外できた件数を表す．

表 6.5: シナリオ判定の混同行列（カットイン）

		抽象空間	
		カットイン	非カットイン
具体空間	カットイン	TP	FN
	非カットイン	FP	TN

表 6.6: 等間隔のグリッドをを用いたカットインシナリオ分析結果

Grid x	Grid y	Scenario	TP	FN	FP	TN
1.00	1.00	Cut-in	3016	0	0	278
1.00	1.00	Cut-out	0	0	0	3294
1.00	1.00	Acceleration	1054	0	0	2240
1.00	1.00	Deceleration	330	0	0	2964
1.75	1.75	Cut-in	3016	0	0	278
1.75	1.75	Cut-out	0	0	0	3294
1.00	1.00	Acceleration	1054	0	0	2240
1.75	1.75	Deceleration	186	144	0	2964
2.00	2.00	Cut-in	3002	14	0	278
2.00	2.00	Cut-out	0	0	0	3294
2.00	2.00	Acceleration	1054	0	0	2240
2.00	2.00	Deceleration	330	0	0	2964
2.65	0.95	Cut-in	3012	4	0	278
2.65	0.95	Cut-out	0	0	0	3294
2.65	0.95	Acceleration	1054	0	0	2240
2.65	0.95	Deceleration	186	144	0	2964
3.50	3.50	Cut-in	1274	1742	0	278
3.50	3.50	Cut-out	0	0	0	3294
3.50	3.50	Acceleration	1054	0	0	2240
3.50	3.50	Deceleration	186	144	0	2964
4.00	4.00	Cut-in	0	3016	0	278
4.00	4.00	Cut-out	0	0	0	3294
4.00	4.00	Acceleration	1054	0	0	2240
4.00	4.00	Deceleration	234	96	0	2964
5.30	1.90	Cut-in	2722	294	0	278
5.30	1.90	Cut-out	0	0	0	3294
5.30	1.90	Acceleration	1054	0	0	2240
5.30	1.90	Deceleration	186	144	0	2964

表 6.7: 等間隔のグリッドを用いたカットアウトシナリオ分析結果

Grid x	Grid y	Scenario	TP	FN	FP	TN
1.00	1.00	Cut-in	0	0	0	4108
1.00	1.00	Cut-out	4108	0	0	0
1.00	1.00	Acceleration	0	0	0	4108
1.00	1.00	Deceleration	1132	0	0	2976
1.75	1.75	Cut-in	0	0	0	4108
1.75	1.75	Cut-out	4108	0	0	0
1.75	1.75	Acceleration	0	0	0	4108
1.75	1.75	Deceleration	796	336	0	2976
2.00	2.00	Cut-in	0	0	0	4108
2.00	2.00	Cut-out	4088	20	0	0
2.00	2.00	Acceleration	0	0	0	4108
2.00	2.00	Deceleration	1132	0	0	2976
2.65	0.95	Cut-in	0	0	0	4108
2.65	0.95	Cut-out	4104	4	0	0
2.65	0.95	Acceleration	0	0	0	4108
2.65	0.95	Deceleration	748	384	0	2976
3.50	3.50	Cut-in	0	0	0	4108
3.50	3.50	Cut-out	1981	2127	0	0
3.50	3.50	Acceleration	0	0	0	4108
3.50	3.50	Deceleration	796	336	0	2976
4.00	4.00	Cut-in	0	0	0	4108
4.00	4.00	Cut-out	0	4108	0	0
4.00	4.00	Acceleration	0	0	0	4108
4.00	4.00	Deceleration	940	192	0	2976
5.30	1.90	Cut-in	0	0	0	4108
5.30	1.90	Cut-out	3970	138	0	0
5.30	1.90	Acceleration	0	0	0	4108
5.30	1.90	Deceleration	748	384	0	2976

表 6.8: 等間隔のグリッドを用いた加速シナリオ分析結果

Grid x	Grid y	Scenario	TP	FN	FP	TN
1.00	1.00	Cut-in	0	0	0	14952
1.00	1.00	Cut-out	0	0	0	14952
1.00	1.00	Acceleration	4287	1901	0	8764
1.00	1.00	Deceleration	0	0	0	14952
1.75	1.75	Cut-in	0	0	0	14952
1.75	1.75	Cut-out	0	0	0	14952
1.75	1.75	Acceleration	3461	2727	0	8764
1.75	1.75	Deceleration	0	0	0	14952
2.00	2.00	Cut-in	0	0	0	14952
2.00	2.00	Cut-out	0	0	0	14952
2.00	2.00	Acceleration	4127	2061	0	8764
2.00	2.00	Deceleration	0	0	0	14952
2.65	0.95	Cut-in	0	0	0	14952
2.65	0.95	Cut-out	0	0	0	14952
2.65	0.95	Acceleration	3385	2803	0	8764
2.65	0.95	Deceleration	0	0	0	14952
3.50	3.50	Cut-in	0	0	0	14952
3.50	3.50	Cut-out	0	0	0	14952
3.50	3.50	Acceleration	3307	2881	0	8764
3.50	3.50	Deceleration	0	0	0	14952
4.00	4.00	Cut-in	0	0	0	14952
4.00	4.00	Cut-out	0	0	0	14952
4.00	4.00	Acceleration	3387	2801	0	8764
4.00	4.00	Deceleration	0	0	0	14952
5.30	1.90	Cut-in	0	0	0	14952
5.30	1.90	Cut-out	0	0	0	14952
5.30	1.90	Acceleration	3088	3100	0	8764
5.30	1.90	Deceleration	0	0	0	14952

表 6.9: 等間隔のグリッドを用いた減速シナリオ分析結果

Grid x	Grid y	Scenario	TP	FN	FP	TN
1.00	1.00	Cut-in	0	0	0	1357
1.00	1.00	Cut-out	0	0	0	1357
1.00	1.00	Acceleration	0	0	0	1357
1.00	1.00	Deceleration	517	120	0	720
1.75	1.75	Cut-in	0	0	0	1357
1.75	1.75	Cut-out	0	0	0	1357
1.75	1.75	Acceleration	0	0	0	1357
1.75	1.75	Deceleration	453	184	0	720
2.00	2.00	Cut-in	0	0	0	1357
2.00	2.00	Cut-out	0	0	0	1357
2.00	2.00	Acceleration	0	0	0	1357
2.00	2.00	Deceleration	517	120	0	720
2.65	0.95	Cut-in	0	0	0	1357
2.65	0.95	Cut-out	0	0	0	1357
2.65	0.95	Acceleration	0	0	0	1357
2.65	0.95	Deceleration	411	226	0	720
3.50	3.50	Cut-in	0	0	0	1357
3.50	3.50	Cut-out	0	0	0	1357
3.50	3.50	Acceleration	0	0	0	1357
3.50	3.50	Deceleration	417	220	0	720
4.00	4.00	Cut-in	0	0	0	1357
4.00	4.00	Cut-out	0	0	0	1357
4.00	4.00	Acceleration	0	0	0	1357
4.00	4.00	Deceleration	381	256	0	720
5.30	1.90	Cut-in	0	0	0	1357
5.30	1.90	Cut-out	0	0	0	1357
5.30	1.90	Acceleration	0	0	0	1357
5.30	1.90	Deceleration	378	259	0	720

表 6.10: 等間隔のグリッドを用いた状態数と圧縮率（カットイン）

Grid x (m)	Grid y (m)	平均状態数 （個）	平均圧縮率 (%)
1.0	1.0	236.5	33.448
1.75	1.75	165.1	55.837
2.0	2.0	145.5	60.667
2.65	0.95	110.5	70.204
3.5	3.5	84.0	77.332
4.0	4.0	73.8	79.947
5.3	1.9	57.1	84.369

表 6.11: 等間隔のグリッドを用いた状態数と圧縮率（カットアウト）

Grid x (m)	Grid y (m)	平均状態数 （個）	平均圧縮率 (%)
1.0	1.0	190.9	40.793
1.75	1.75	130.3	62.059
2.0	2.0	116.5	65.191
2.65	0.95	87.5	74.271
3.5	3.5	66.5	80.354
4.0	4.0	59.3	82.084
5.3	1.9	45.3	86.418

表 6.12: 等間隔のグリッドを用いた状態数と圧縮率（加速）

Grid x (m)	Grid y (m)	平均状態数 (個)	平均圧縮率 (%)
1.0	1.0	214.8	51.552
1.75	1.75	138.6	69.696
2.0	2.0	121.4	73.364
2.65	0.95	92.1	79.767
3.5	3.5	70.1	84.517
4.0	4.0	61.5	86.387
5.3	1.9	46.9	89.553

表 6.13: 等間隔のグリッドを用いた状態数と圧縮率（減速）

Grid x (m)	Grid y (m)	平均状態数 （個）	平均圧縮率 (%)
1.0	1.0	73.4	53.514
1.75	1.75	43.1	71.99
2.0	2.0	40.4	74.559
2.65	0.95	29.0	81.194
3.5	3.5	22.3	85.521
4.0	4.0	20.6	86.948
5.3	1.9	15.3	90.08

結果として，以下の傾向が確認された．

FN の発生傾向: 表 6.6 から表 6.9 に示すように，グリッド幅が小さい（2.00 m 以下）範囲では FN はほぼ発生せず，高い再現率を維持している．しかし，グリッド幅が車両サイズに近づく（3.50 m 以上）と，FN が急激に増加する傾向が見られた．

FP の非発生: 全てのシナリオにおいて，FP は発生しなかった．

圧縮率と精度のトレードオフ: 表 6.10 から表 6.13 に示すように，グリッド幅を大きくすることで圧縮率は 30%台から 90%台まで向上する．

安全性判定の精度に関する実験結果　安全性検証における判定精度の結果を表 6.15 に示す．ここで，Rule Type における ACC は相対加速度に基づく危険判定ルール，VEL は相対速度に基づく危険判定ルールを表す．

表 6.14: 安全性判定の混同行列

		抽象空間	
		安全	危険
具体空間	安全	TP	FN
	危険	FP	TN

表 6.15: 各シナリオにおける安全性分析結果（全グリッドサイズ共通）

Scenario	Rule Type	TP	FN	FP	TN
Cut-in	ACC	2	3286	0	6
	VEL	2	340	0	2952
Cut-out	ACC	56	4052	0	0
	VEL	56	1172	0	2880
Acceleration	ACC	96	14856	0	0
	VEL	96	7921	0	6935
Deceleration	ACC	16	1341	0	0
	VEL	16	821	0	520

本実験では，グリッドサイズの変化が判定結果（TP/FP/FN/TN の件数）に影響を与えなかった．

結果の詳細を見ると，FN は 0 件であり，提案手法は健全性を維持していることが確認できた．一方で，FP については，相対加速度（ACC）ルールの方が多く発生している．相対速度（VEL）を用いたルールでは，FP の発生率が相対加速度（ACC）の場合と比較して 10%〜60%程度抑制されている．

6.2 車両周辺領域を分割したデータマッピング実験

目的 前節の「等間隔グリッド」が空間を均一に分割する手法であったのに対し，本節では自車（Ego）を中心とした相対的な位置関係に基づく抽象化手法について評価を行う．具体的には，車両周辺を意味的な領域（前方，側方，後方など）に分割する手法を適用し，シナリオの識別精度，状態数の圧縮率などに変化が出るかを確かめるために実験を行う．

データセット 6.1.1 節で構築したデータセットと同一のものを使用する．

6.2.1 実験方法

領域分割の設定 本実験では，第 4.3 節で定義した 2 種類の領域分割モデル，すなわち「9 領域モデル」および「15 領域モデル」を用いて抽象化を行う．

まず，9 領域モデルは，自車位置を中心として周囲を 8 近傍（前後左右および斜め方向）に分割することで，合計 9 つの空間領域を定義するものである．一方，15 領域モデルは，この 9 領域構成を基礎としつつ，前方と後方の領域を分割することで，より詳細な空間分割を実現した計 15 領域のモデルである．

実験手順は 6.1.2 節と同様である．ただし，本手法では領域の形状が自車位置に基づいて一意に決定されるため，グリッド幅のような可変パラメータは存在しない．したがって，上記 2 つのモデルそれぞれについて，全シナリオデータの抽象化および判定処理を行った．

6.2.2 実験結果

シナリオ検出の精度および状態数の圧縮に関する実験結果 9 領域モデルおよび 15 領域モデルを用いた場合の分析結果を表 6.16 から表 6.19 に示す．表 6.16, 表 6.17 は各シナリオにおける混同行列を，表 6.18, 表 6.19 はそれぞれの平均の状態数と状態数の圧縮率を示している．

表 6.16: 9 領域の分析結果

Target Scenario	Detected Scenario	TP	FN	FP	TN
Cut-in	Cut-in	3016	0	0	278
	Cut-out	0	0	0	3294
	Acceleration	1054	0	0	2240
	Deceleration	186	144	0	2964
Cut-out	Cut-in	0	0	0	4108
	Cut-out	4108	0	0	0
	Acceleration	0	0	0	4108
	Deceleration	748	384	0	2976
Acceleration	Cut-in	0	0	0	14952
	Cut-out	0	0	0	14952
	Acceleration	3572	2616	0	8764
	Deceleration	0	0	0	14952
Deceleration	Cut-in	0	0	0	1357
	Cut-out	0	0	0	1357
	Acceleration	0	0	0	1357
	Deceleration	477	160	0	720

表 6.17: 15 領域の分析結果

Target Scenario	Detected Scenario	TP	FN	FP	TN
Cut-in	Cut-in	2990	26	0	278
	Cut-out	0	0	0	3294
	Acceleration	1054	0	0	2240
	Deceleration	186	144	0	2964
Cut-out	Cut-in	0	0	0	4108
	Cut-out	4092	16	0	0
	Acceleration	0	0	0	4108
	Deceleration	748	384	0	2976
Acceleration	Cut-in	0	0	0	14952
	Cut-out	0	0	0	14952
	Acceleration	3572	2616	0	8764
	Deceleration	0	0	0	14952
Deceleration	Cut-in	0	0	0	1357
	Cut-out	0	0	0	1357
	Acceleration	0	0	0	1357
	Deceleration	477	160	0	720

表 6.18: 9 領域の状態数と圧縮率

Scenario	平均状態数	平均圧縮率 (%)
Cut-in	2.8	98.8
Cut-out	2.2	98.9
Acceleration	1.5	99.1
Deceleration	1.6	99.4

表 6.19: 15 領域の状態数と圧縮率

Scenario	平均状態数	平均圧縮率 (%)
Cut-in	3.9	98.4
Cut-out	2.7	98.6
Acceleration	2.6	98.6
Deceleration	2.1	99.2

両モデルともに平均状態数は 4 個以下となり，圧縮率は 98%を超える値を示した．検出精度に関しては，シナリオを高い精度で特定できていることが確認された．

安全性判定の分析結果　安全性検証における判定精度の結果を表6.20および表6.21に示す.

表 6.20: 9 領域での安全性評価

Scenario	TP	FN	FP	TN
Cut-in	2	2572	0	720
Cut-out	56	2612	0	1440
Acceleration	96	7921	0	6935
Deceleration	16	821	0	520

表 6.21: 15 領域での安全性評価

Scenario	TP	FN	FP	TN
Cut-in	2	1852	0	1440
Cut-out	56	1172	0	2880
Acceleration	96	7921	0	6935
Deceleration	16	821	0	520

全てのシナリオおよび両モデルにおいて，FN は 0 件であった．5 領域モデルを用いることで，カットインおよびカットアウトシナリオにおける TN が増加（FP が減少）した.

6.3　実際のシミュレータ軌道での実験

目的　前節まではパラメータを規則的に変化させた疑似データセットを用いていたが，本節ではより現実的な環境における手法の有効性を検証する．そのために，オープンソースの自動運転シミュレータである AWSIM [31, 32] 上で取得された走行軌道データに対し，提案手法を適用する．疑似的に生成した軌道と異なり，車両挙動にノイズや非線形な動きが含まれるため，実際の開発環境に近い条件下での評価を行うことができる.

実験方法　使用するデータセットは，文献 [31, 32] で公開されているシナリオデータセットである．このデータセットでは，Autoware と AWSIM を連携し，AWSIM-Script を用いて自車と他車の動きを定義し動作させることで，yaml または，json 形式の車両軌道データを取得できる．本スクリプトでは，まず車両の発生位置やゴール地点をレーン ID（例：`"TrafficLane.112"`）によって定義している．変数 `route1` は NPC の走行計画を表すリストであり，`cut-in` 関数を用いることで，隣接車線からの割り込み動作を明示的に記述している．ここで，`npc1` および `ego` の定義におい

て，車種（"hatchback"等），初期位置，および目標速度や加速度（max-velocity, acceleration）がパラメータとして与えられている．最終的に run 関数を実行することで，これらの定義に基づいたシミュレーションが開始され，指定されたタイムアウト時間（saving-timeout）まで軌道データが記録される．実際の AWSIM-Script の例をソースコード 6.2 に示す．このスクリプトは AWSIM 上で他車（NPC）が自車（Ego）の前方に割り込む「カットインシナリオ」を生成するためのスクリプトである．

ソースコード 6.2: AWSIM-Script の例（[31] より引用）

```
spawnPos = "TrafficLane.112";
goal = "TrafficLane.111" at 180;
npcspeed = 2.778;
route1 = [
    "TrafficLane.112" max-velocity(npcspeed),
    cut-in(_, npcspeed, 0.6, 9),
    "TrafficLane.111" max-velocity(npcspeed)
];

npc1 = NPC("hatchback", _, goal, route1, [acceleration(20)]);

ego = Ego(spawnPos left 0, "TrafficLane.111" at 130, [max-velocity
    (5.556)]);

run(ego, [npc1], [saving-timeout(70)]);
```

抽象化手法には，前節の実験において FN が発生せず，かつ TN の精度が最も高かったことから，15 領域分割モデルを採用する．評価項目はこれまでと同様に，シナリオ検出の混同行列および状態数の圧縮率とする．

シナリオ検出の精度および状態数の圧縮に関する実験結果　実験結果を表 6.22 および表 6.23 に示す．なお，表中の Scenario 名はデータセットのフォルダ名 [31, 32] に準拠している．

表 6.22: シミュレーションデータによる各シナリオの混同行列

Scenario	Target	TP	FN	FP	TN
cutin/40-20	cutin	34	0	0	0
	cutout	12	0	0	22
	accel	0	0	0	34
	decel	34	0	0	0
cutin/40-30	cutin	30	0	0	0
	cutout	16	1	0	13
	accel	0	0	0	30
	decel	30	0	0	0
cutout	cutin	0	0	0	1381
	cutout	30	0	0	1351
	accel	0	42	0	1339
	decel	1381	0	0	0
deceleration/lidarcamyaml	cutin	0	0	0	468
	cutout	0	0	0	468
	accel	0	0	0	468
	decel	468	0	0	0
deceleration/yaml	cutin	0	0	0	1254
	cutout	0	0	0	1254
	accel	0	0	0	1254
	decel	1253	1	0	0
swerve	cutin	0	0	0	9
	cutout	0	0	0	9
	accel	0	0	0	9
	decel	9	0	0	0
uturn	cutin	10	0	0	0
	cutout	0	0	0	10
	accel	0	0	0	10
	decel	10	0	0	0

表 6.23: シミュレーションデータによるシナリオ別の平均状態数と圧縮率

Scenario	平均状態数 (個)	平均圧縮率 (%)
cutin/40-20	12.2	99.524
cutin/40-30	13.2	99.411
cutout	12.9	99.395
deceleration/lidarcamyaml	11.1	99.463
deceleration/yaml	10.3	99.518
swerve	8.1	99.658
uturn	12.8	99.416

基本シナリオの検出精度: ‘cutin‘ や ‘cutout‘ といった基本的なシナリオにおいて，高い TP（True Positive）を示した．特に ‘cutin‘ シナリオにおいては，同時に ‘decel‘（減速）も検出されている．これは，割り込み動作に伴い自車または他車が減速調整を行うという物理的な因果関係を正しく捉えているといえる．

また，集計上は同じように記録される軌道であっても，カットインしている位置や，減速のタイミングが異なる場合，本手法では異なる抽象軌道が生成される．減速と同時にカットインが起こる軌道を図 6.1, 減速した後にカットインが起こる軌道を図 6.2 に示す．

Time	抽象空間	備考	
1(1-62)	far-right		
2(63-547)	far-right	【減速】	
3(548-751)	lead-right	【減速】	
4(752-773)	lead	【減速】	【カットイン】
5(774-776)	lead		
6(777-1053)	lead		
7(1054-1379)	far-front		
8(1380-1384)	far-front		
9(1385-1877)	far-front	【減速】	
10(1878)	far-front		
11(1879-2848)	far-front		

図 6.1: 減速とともにカットインする軌道

Time	抽象空間	備考
1(1-1085)	far-right	
2(1086-1106)	far-right	【減速】
3(1107-1116)	far-right	
4(1117-1596)	far-right	【減速】
5(1597-1811)	lead-right	【減速】
6(1812-1815)	lead-right	
7(1816-1888)	lead-right	
8(1889-2147)	lead	【カットイン】
9(2148-2627)	far-front	
10(2628-2654)	far-front	

図 6.2: 減速した後にカットインする軌道

未定義シナリオの挙動: 本手法で定義していない ‘swerve‘（蛇行）や ‘uturn‘（U ターン）のデータに対して，‘swerve‘ に対しては ‘accel‘の反応が見られた．また，‘uturn‘ が ‘cutin‘ および ‘decel‘ として検出されたのは，U ターン動作が「対向車線への進入（Cut-in 的動作）」と「減速」の組み合わせで構成されていること示している．

状態数と圧縮率: 表 6.23 に示すように，状態数に関してはどのシナリオにおいても 10 個前後であるとの結果が得られた，圧縮率に関しては，実際のシミュレータ軌道においても 99%以上の圧縮率であった．

6.4 Lanelet2 と STL を用いた手法との比較実験

目的 本研究で提案した抽象化手法の有効性を相対的に評価するために，形式手法の一つである STL（Signal Temporal Logic: 信号時相論理）を用いたシナリオ検出手法 [17] との比較実験を行う．本実験では，地図情報の誤差や定義の揺らぎに対するロバスト性を検証するために，この判定において車線逸脱や進入の閾値となる「車線幅（Road width）」を意図的に変化させた際の，検出精度の推移を評価する．

実験方法 使用するデータセットは，第 6.1.1 節で生成したデータセットと同一のものを用いる．本実験では，比較手法における車線幅パラメータ W を 1.9 m から 8.0 m の範囲で段階的に変化させ，各設定における判定結果を取得した．比較対象の手法は，高精度地図（Lanelet2）上の車線情報と車両の軌道データに対し，時相論理式を適用して判定を行うものである．

本実験で用いた定義を 6.1 に示す．

定義 6.1 (stl における定義).

カットイン 以下の 2 つの条件が共に真となる場合として定義する．

1. **自車のレーン維持**: 全ての時刻において，自車が初期の走行車線を維持していること．
2. **割り込みの発生**: 将来のある時刻において，他車が隣接レーンから自車と同一のレーンに移動し，かつその時点で「他車の後端 $>$ 自車の先端」という前後関係が成立すること．

カットアウト 以下の 3 つの条件が共に真となる場合として定義する．

1. **初期位置**: 初期の時刻において，他車と自車が同一レーンに存在し，かつ「他車の後端 $>$ 自車の先端（他車が前方）」の位置関係にあること．
2. **自車のレーン維持**: 全ての時刻において，自車が初期の走行車線を維持していること．
3. **逸脱の発生**: 将来のある時刻において，他車が自車線から隣接レーンへ移動すること．

加速（後続車の接近） 以下の 3 つの条件が共に真となる場合として定義する．

1. **初期位置**: 初期の時刻において，他車と自車が同一レーンに存在し，かつ「自車の後端 $>$ 他車の先端（他車が後方）」の位置関係にあること．
2. **他車のレーン維持**: 全ての時刻において，他車が初期の走行車線を維持していること．
3. **加速による接近**: 他車の速度が自車の速度を上回る状態（$v_{npc} > v_{ego}$）となり，車間距離が短縮すること．

減速（先行車の減速） 以下の 3 つの条件が共に真となる場合として定義する．

1. **初期位置**: 初期の時刻において，他車と自車が同一レーンに存在し，かつ「他車の後端 $>$ 自車の先端（他車が前方）」の位置関係にあること．
2. **他車のレーン維持**: 全ての時刻において，他車が初期の走行車線を維持していること．
3. **減速による接近**: 他車の速度が自車の速度を下回る状態（$v_{npc} < v_{ego}$）となり，車間距離が短縮すること．

評価指標は，シナリオ検出の混同行列（TP, FN, FP, TN）とする．なお，比較対象手法は論理式による真偽判定を行うものであり，状態遷移としての抽象化を行わないため，本実験では「状態数の圧縮率」は評価対象外とし,「シナリオ識別精度」のみを比較する．

実験結果　車線幅を 1.9 m から 8.0 m まで変化させた際の各シナリオの判定結果を表 6.24 から表 6.27 に示す.

表 6.24: Lanelet2 と STL によるカットインシナリオ分析結果

Road width	Scenario	TP	FN	FP	TN
1.9m	Cut-in	3016	0	0	278
1.9m	Cut-out	0	0	0	3294
1.9m	Acceleration	0	1054	0	2240
1.9m	Deceleration	0	330	0	2964
2.0m	Cut-in	3016	0	0	278
2.0m	Cut-out	0	0	0	3294
2.0m	Acceleration	0	1054	0	2240
2.0m	Deceleration	0	330	0	2964
3.5m	Cut-in	3016	0	0	278
3.5m	Cut-out	0	0	0	3294
3.5m	Acceleration	0	1054	0	2240
3.5m	Deceleration	0	330	0	2964
3.8m	Cut-in	3016	0	0	278
3.8m	Cut-out	0	0	0	3294
3.8m	Acceleration	0	1054	0	2240
3.8m	Deceleration	0	330	0	2964
4.0m	Cut-in	3006	10	0	278
4.0m	Cut-out	0	0	0	3294
4.0m	Acceleration	0	1054	0	2240
4.0m	Deceleration	0	330	0	2964
7.0m	Cut-in	1361	1655	1	277
7.0m	Cut-out	0	0	0	3294
7.0m	Acceleration	527	527	0	2240
7.0m	Deceleration	164	166	48	2916
8.0m	Cut-in	0	3016	0	278
8.0m	Cut-out	0	0	0	3294
8.0m	Acceleration	1054	0	0	2240
8.0m	Deceleration	328	2	96	2868

表 6.25: Lanelet2 と STL によるカットアウトシナリオ分析結果

Road width	Scenario	TP	FN	FP	TN
1.9m	Cut-in	0	0	0	4108
1.9m	Cut-out	4108	0	0	0
1.9m	Acceleration	0	0	0	4108
1.9m	Deceleration	0	1132	0	2976
2.0m	Cut-in	0	0	0	4108
2.0m	Cut-out	4108	0	0	0
2.0m	Acceleration	0	0	0	4108
2.0m	Deceleration	0	1132	0	2976
3.5m	Cut-in	0	0	0	4108
3.5m	Cut-out	4108	0	0	0
3.5m	Acceleration	0	0	0	4108
3.5m	Deceleration	0	1132	0	2976
3.8m	Cut-in	0	0	0	4108
3.8m	Cut-out	4108	0	0	0
3.8m	Acceleration	0	0	0	4108
3.8m	Deceleration	0	1132	0	2976
4.0m	Cut-in	0	0	0	4108
4.0m	Cut-out	4108	0	0	0
4.0m	Acceleration	0	0	0	4108
4.0m	Deceleration	0	1132	0	2976
7.0m	Cut-in	0	0	0	4108
7.0m	Cut-out	2054	2054	0	0
7.0m	Acceleration	0	0	0	4108
7.0m	Deceleration	565	567	48	2928
8.0m	Cut-in	0	0	0	4108
8.0m	Cut-out	0	4108	0	0
8.0m	Acceleration	0	0	0	4108
8.0m	Deceleration	1130	2	96	2880

表 6.26: Lanelet2 と STL による加速シナリオ分析結果

Road width	Scenario	TP	FN	FP	TN
1.9m	Cut-in	0	0	0	14952
1.9m	Cut-out	0	0	0	14952
1.9m	Acceleration	6188	0	1435	7329
1.9m	Deceleration	0	0	0	14952
2.0m	Cut-in	0	0	0	14952
2.0m	Cut-out	0	0	0	14952
2.0m	Acceleration	6188	0	1435	7329
2.0m	Deceleration	0	0	0	14952
3.5m	Cut-in	0	0	0	14952
3.5m	Cut-out	0	0	0	14952
3.5m	Acceleration	6188	0	1435	7329
3.5m	Deceleration	0	0	0	14952
3.8m	Cut-in	0	0	0	14952
3.8m	Cut-out	0	0	0	14952
3.8m	Acceleration	6188	0	1435	7329
3.8m	Deceleration	0	0	0	14952
4.0m	Cut-in	0	0	0	14952
4.0m	Cut-out	0	0	0	14952
4.0m	Acceleration	6188	0	1435	7329
4.0m	Deceleration	0	0	0	14952
7.0m	Cut-in	0	0	0	14952
7.0m	Cut-out	0	0	0	14952
7.0m	Acceleration	6188	0	1435	7329
7.0m	Deceleration	0	0	0	14952
8.0m	Cut-in	0	0	0	14952
8.0m	Cut-out	0	0	0	14952
8.0m	Acceleration	6188	0	1435	7329
8.0m	Deceleration	0	0	0	14952

表 6.27: Lanelet2 と STL による減速シナリオ分析結果

Road width	Scenario	TP	FN	FP	TN
1.9m	Cut-in	0	0	0	1357
1.9m	Cut-out	0	0	0	1357
1.9m	Acceleration	0	0	0	1357
1.9m	Deceleration	637	0	80	640
2.0m	Cut-in	0	0	0	1357
2.0m	Cut-out	0	0	0	1357
2.0m	Acceleration	0	0	0	1357
2.0m	Deceleration	637	0	80	640
3.5m	Cut-in	0	0	0	1357
3.5m	Cut-out	0	0	0	1357
3.5m	Acceleration	0	0	0	1357
3.5m	Deceleration	637	0	80	640
3.8m	Cut-in	0	0	0	1357
3.8m	Cut-out	0	0	0	1357
3.8m	Acceleration	0	0	0	1357
3.8m	Deceleration	637	0	80	640
4.0m	Cut-in	0	0	0	1357
4.0m	Cut-out	0	0	0	1357
4.0m	Acceleration	0	0	0	1357
4.0m	Deceleration	637	0	80	640
7.0m	Cut-in	0	0	0	1357
7.0m	Cut-out	0	0	0	1357
7.0m	Acceleration	0	0	0	1357
7.0m	Deceleration	637	0	80	640
8.0m	Cut-in	0	0	0	1357
8.0m	Cut-out	0	0	0	1357
8.0m	Acceleration	0	0	0	1357
8.0m	Deceleration	637	0	80	640

表 6.24 および表 6.25 に示すように，車線幅が 4.0 m 程度までの範囲では，高い精度でシナリオを検出できている．しかし，車線幅を 7.0 m 以上に設定すると，FN（見逃し）が増加し，検出精度が著しく低下した．

表 6.26 および表 6.27 に示すように，加速・減速シナリオに関しては，車線幅の設定にかかわらず同一の結果となった．これは，縦方向の挙動判定が主に速度や加速度の閾値に依存しており，車線幅という空間的なパラメータの影響を受けないためである．

6.5 LLMとの比較

目的　本項では，数値時系列データである車両軌道に対するLLMの処理能力を検証し，提案手法の出力結果との対比を行う．具体的には，同一の軌道データに対して「定性的な説明（状況記述）」と「定量的な判定（シナリオ分類）」の2種類の観点で比較を行い，その出力形式および内容を確認する．

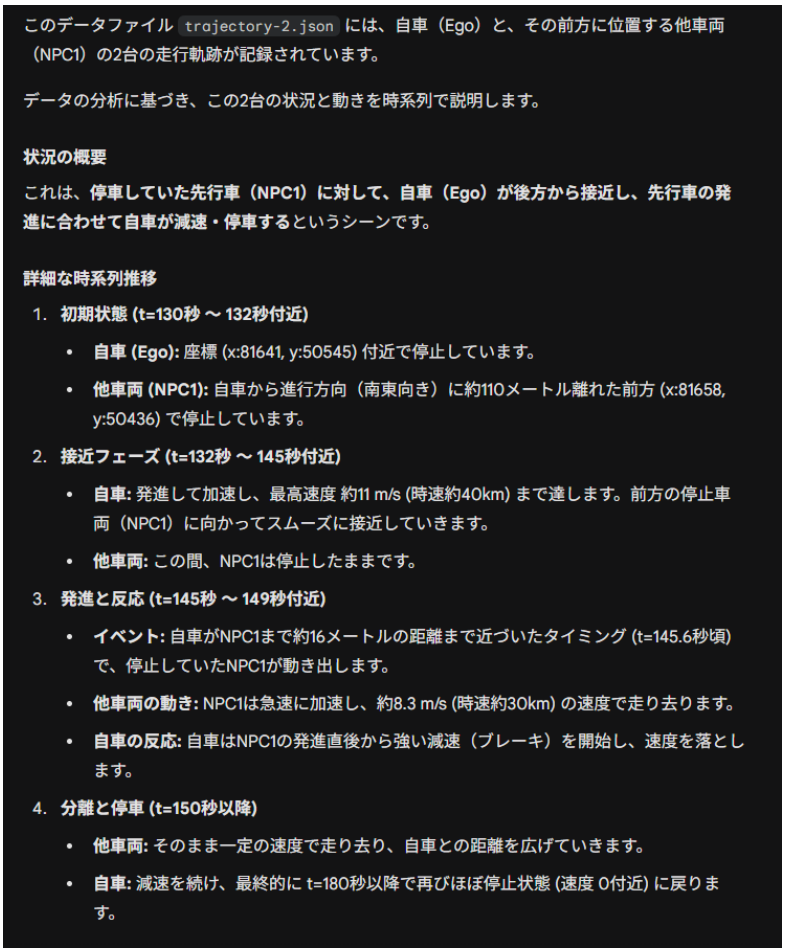
実験方法　使用モデルには **Gemini 3 Pro** を用いた．モデルのパラメータはデフォルト設定とし，プロンプトには指示文に加え，シミュレータから出力されたAWSIMの軌道データ（JSON形式のテキスト）を入力した．入力したシナリオはカットインしている軌道を用いた．

ケース1：2台の車両であることを明記した場合

プロンプト

> この軌道は2台の車両の軌道です．この軌道の状況に関して説明してください．

出力結果　LLMの出力結果を図6.3に示す．図に示すように，LLMは入力された数値データの推移に基づき，車両の挙動や位置関係を自然言語の文章として出力した．出力内には「スムーズに接近」「強い減速」といった挙動に関する記述が含まれていることが確認された．



このデータファイル `trajectory-2.json` には、自車（Ego）と、その前方に位置する他車両（NPC1）の2台の走行軌跡が記録されています。

データの分析に基づき、この2台の状況と動きを時系列で説明します。

状況の概要

これは、**停車していた先行車（NPC1）に対して、自車（Ego）が後方から接近し、先行車の発進に合わせて自車が減速・停車する**というシーンです。

詳細な時系列推移

1. **初期状態 (t=130秒 ～ 132秒付近)**
   - **自車 (Ego):** 座標 (x:81641, y:50545) 付近で停止しています。
   - **他車両 (NPC1):** 自車から進行方向（南東向き）に約110メートル離れた前方 (x:81658, y:50436) で停止しています。
2. **接近フェーズ (t=132秒 ～ 145秒付近)**
   - **自車:** 発進して加速し、最高速度 約11 m/s (時速約40km) まで達します。前方の停止車両（NPC1）に向かってスムーズに接近していきます。
   - **他車両:** この間、NPC1は停止したままです。
3. **発進と反応 (t=145秒 ～ 149秒付近)**
   - **イベント:** 自車がNPC1まで約16メートルの距離まで近づいたタイミング (t=145.6秒頃) で、停止していたNPC1が動き出します。
   - **他車両の動き:** NPC1は急速に加速し、約8.3 m/s (時速約30km) の速度で走り去ります。
   - **自車の反応:** 自車はNPC1の発進直後から強い減速（ブレーキ）を開始し、速度を落とします。
4. **分離と停車 (t=150秒以降)**
   - **他車両:** そのまま一定の速度で走り去り、自車との距離を広げていきます。
   - **自車:** 減速を続け、最終的に t=180秒以降で再びほぼ停止状態 (速度 0付近) に戻ります。

図 6.3: 軌道データに対し自由記述での状況説明を求めた場合の LLM 出力

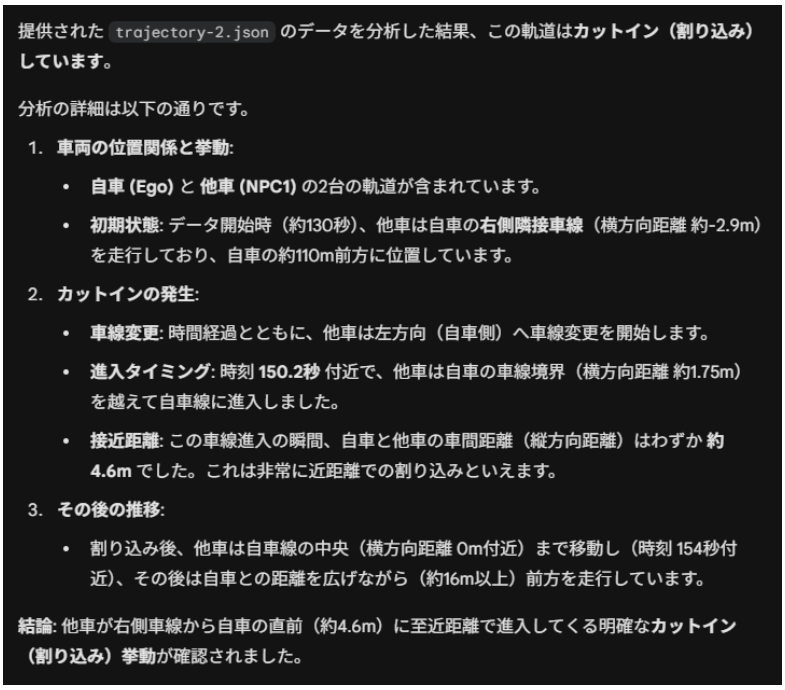
ケース 2：カットインしているかを聞いた場合

次に，対象がカットインシナリオであることを前提とし，その正否を判定させるタスクを行った．

プロンプト

この軌道は 2 台の車両の軌道です．この軌道はカットインしていますか?

出力結果　LLM の出力結果を図 6.4 に示す．横方向距離に着目して，カットインを検出している．進入タイミングを秒数で説明したり，接近情報で最も近づいた場合について自然言語で記述している．



提供された `trajectory-2.json` のデータを分析した結果、この軌道は**カットイン（割り込み）しています。**

分析の詳細は以下の通りです。

1. **車両の位置関係と挙動:**
   - **自車 (Ego)** と **他車 (NPC1)** の2台の軌道が含まれています。
   - **初期状態:** データ開始時（約130秒）、他車は自車の**右側隣接車線**（横方向距離 約-2.9m）を走行しており、自車の約110m前方に位置しています。
2. **カットインの発生:**
   - **車線変更:** 時間経過とともに、他車は左方向（自車側）へ車線変更を開始します。
   - **進入タイミング:** 時刻 **150.2秒** 付近で、他車は自車の車線境界（横方向距離 約1.75m）を越えて自車線に進入しました。
   - **接近距離:** この車線進入の瞬間、自車と他車の車間距離（縦方向距離）はわずか **約4.6m** でした。これは非常に近距離での割り込みといえます。
3. **その後の推移:**
   - 割り込み後、他車は自車線の中央（横方向距離 0m付近）まで移動し（時刻 154秒付近）、その後は自車との距離を広げながら（約16m以上）前方を走行しています。

結論: 他車が右側車線から自車の直前（約4.6m）に至近距離で進入してくる明確な**カットイン（割り込み）挙動**が確認されました。

図 6.4: 車両軌道がカットインしているかを LLM に聞いた場合

提案手法による出力結果　比較対象として，同一の軌道データを提案手法（グリッド/領域分割）で処理した結果を図 6.5 に示す．提案手法の出力は，領域の遷移を lead-right,lead のような表現を列として出力している．

Time	抽象空間	備考
1(1-62)	far-right	
2(63-520)	far-right	【減速】
3(521-768)	lead-right	【減速】
4(769-771)	lead-right	
5(772-865)	lead-right	
6(866-1164)	lead	【カットイン】
7(1165-1170)	lead	
8(1171-1567)	lead	【減速】
9(1568-1768)	lead	
10(1769-1940)	lead	
11(1941-2553)	far-front	

図 6.5: 提案手法により同一シナリオを抽象化した結果

大量データ処理時の挙動　約 4000 件の軌道データを一括で入力した場合の応答を図 6.6 に示す．コンテキスト長（トークン数）の制限に関するエラーメッセージが返され，解析処理は実行されなかった．

添付されたコードに含まれるファイルが多すぎるため処理できません。詳細は
https://support.google.com/gemini?p=code_limit をご確認ください。

図 6.6: 大量のファイルを処理させた際に発生したエラーの例

第7章　考察

実験結果に基づき，提案手法ごとの精度，特性について述べる．また，提案手法における利点や限界についても考察を行う．

7.1　等間隔のグリッドを用いたデータマッピングの実験結果の考察

7.1.1　シナリオの識別精度

区間の幅（グリッドサイズ）が事象検知の性能に与える影響を評価する．どのシナリオにおいても，FP が発生していないため，抽象空間でカットインと判断された軌道は具体空間でもカットインしているといえる．区間の幅を大きくしていくと，検知の精度が低下する傾向が見られる．これは，カットインの定義をグリッドをまたぐというイベントをもとに判断しているため，グリッドが粗くなると実際には車が移動していてもグリッド内にとどまっているように抽象軌道では判断される．これにより，隣接車線への移動を正確に捕捉できなくなるためである．今回生成した軌道ではシミュレーションの開始地点を $\pm 3.5\,\mathrm{m}, 0\,\mathrm{m}$ としているため 3.5m 以上の区間幅にするとカットインとカットアウトの見逃しが増えていることが読み取れる．したがって，カットインのような繊細な横方向の挙動を検出するためには，車線幅よりも細かいグリッド設定（例: 車線幅の半分等）が推奨される．

一方で，加速，減速シナリオにおいては，グリッドサイズを $1.0\,\mathrm{m}$ から $5.3\,\mathrm{m}$ まで変化させても，識別結果（TP, FP など）に大きな変動は見られなかった．この理由としては，今回の軌道では，短い時間の間に加速，減速が発生しているケースが少なく，長時間にわたって一定の加速度が維持されている軌道が場合が多かったためと考えられる．そのため，加速減速を交互に繰り返すような複雑な軌道であれば，グリッドサイズの影響がより顕著に現れる可能性がある．今回の抽象化の狙いとしては，細かい車両の状態を知るよりも，シナリオ全体の流れが分かるような抽象的な軌道を生成することが目標であるため，この点において，目的に沿った適切な抽象化が達成できていると考える．

7.1.2　シナリオの状態数と圧縮率

生成された抽象軌道の可読性についても評価する． 表 4.2 で示したように，状態数が 10 個程度であれば，人間が目視でシナリオを容易に理解・分析することができる． しかし，固定幅のグリッドを用いた等間隔データマッピングでは，元の軌道の長さや車両速度によって生成される状態数が大きく変動する．

実験結果の表 6.10 を確認すると，グリッドサイズを大きくすることで平均状態数は大幅に減少し，圧縮率は向上することが確認された．例えば，グリッドサイズを 1.0 m から 3.5 m に変更することで，状態数を約 1/3 に削減できた．一方で，1.0, m 四方のグリッドではデータ量は半分程度に削減されたものの，依然として状態数は 100〜200 個に達しており，一見して挙動を分析することは困難である（表 4.3）．

一方，今回の実験で最も状態数を削減できた $w_x = 5.3, w_y = 1.9$ の設定では，可読性は向上したものの， $w_x = 1.0, w_y = 1.0$ の場合と比較して，カットインなどをを見逃すケースが増加した．以上の結果から,「可読性の向上（圧縮率)」と「検出精度」の間にトレードオフの関係が存在することが明らかとなった．対象とする交通シナリオに応じて適切なグリッド解像度を選択する必要があるが，固定幅のグリッドを用いる限り，多様なシナリオに対して常に最適な可読性と精度を両立させることは困難である．したがって，自車からの距離によってグリッド幅を変える抽象化手法や，物理的な距離に依存しない抽象化手法の導入が求められる．

7.2　車両周辺の領域を分割した手法の実験結果の考察

7.2.1　シナリオの識別精度

結果から，提案手法は「カットイン」および「カットアウト」の認識において高い精度（高い TP，低い FN）を示した．これは，これらのシナリオが「車線（横方向）の移動」を定義としており，提案手法の領域分割（Left/Center/Right）がその境界を明確に捉えているためである．一方で,「加速」「減速」シナリオに関しては，グリッド手法と比較して検出漏れ（FN）が増加する傾向が見られた．これは，提案手法が空間を広く分割しているため，同一領域内（例えば lead 領域内）での加減速による位置変化が，抽象状態の遷移として現れにくいためであると考えられる．また，9 領域と 15 領域（Far 領域の追加）を比較すると，15 領域の方が FN がわずかに改善される傾向が見られた．これは遠方（Far）と近傍の区別がついたことで，接近挙動の一部をより詳細に捉えられたためと推測される．

自車周辺の領域分割（9 領域および 15 領域）を用いた手法は，等間隔のグリッドを用いた手法と比較して，状態数の変動が少なく，可読性が向上した．特筆すべきは，生成される状態数が元の軌道の長さや車両速度に依存せず，ほとんどのシナリオにおいて平均 10 個以内の状態数（実験結果では平均 2.8 状態）に収束し

た点である．本手法によって生成された数ステップの遷移系列は，人間が一目で全体像を把握し，ストーリーとして理解できる範囲に収まっている．

さらに重要な点は，情報の「質」の変化である．グリッド手法が座標を離散化しただけの「量的抽象化（Metric Abstraction）」であったのに対し，提案手法は‘lead-right‘（右前方）や ‘cut-in‘（割り込み）といった，JAMA フレームワーク等の仕様書や人間が自然言語で用いる概念と 1 対 1 で対応する「質的抽象化（Semantic Abstraction）」を実現している．この「意味的ギャップ（Semantic Gap）」の解消こそが，単なるデータの圧縮以上の本質的な貢献であり，専門家でなくとも車両の挙動を即座に理解できる可読性の高さを裏付けている．

7.2.2　シナリオの状態数と圧縮率

等間隔のデータマッピングの手法と比較して平均の状態数が減少していることが表 6.10, 表 6.18, 表 6.19 から分かる．また，図 6.1, 図 6.2 でも示したように，個別のシナリオを確認してもどこでカットインや加速が発生しているかが分かりやすい．

等間隔のグリッドベースの手法では，抽象軌道が表 4.3 のような数値的なインデックスの羅列となり，その意味を人間が直感的に理解することは困難であった．一方，車両周辺の領域を分割した手法では，軌道が「follow-left → lead-left → lead」のように記述される．これは JAMA フレームワーク等のシナリオ記述と親和性が高く，車両軌道を目で見て分析しやすい形であるといえる．図 6.1, 図 6.2 の結果が示すように，数百ステップにおよぶ時系列データが，わずか数ステップの意味のある遷移に要約されており,「抽象度のギャップ」を埋めるという本研究の目的に対して極めて有効であると評価できる．表 6.18 および表 6.19 に示す通り，提案手法の平均状態数は約 2.0〜3.0 程度であり，圧縮率は約 1.0%という極めて高い値を示した．等間隔のグリッドベースの手法（圧縮率 10%〜50%程度）と比較しても，データ量は大きく削減されている．これは，人が見て分析をするという点においても利点となる．

7.2.3　9 領域から 15 領域に拡張した場合の効果

この縦方向運動の検知遅れに対し，15 領域分割において導入した「Far（遠方）領域」は有効に機能した．Far 領域を設定することで，車両が遠方から近傍へと接近するプロセスを ‘far-rear‘ → ‘follow‘ のような状態遷移として捉えることが可能となる．これにより，抽象軌道を抜き出して確認した際に，どのあたりからカットインが発生したかなど，より詳細な状況に関して，分析することが可能になった．

どの程度の加減速を検知すべきかは目的によって変わるが，本研究のように「自車周辺に近づいて危険となる可能性があるシナリオ」を可視化・スクリーニングする目的においては，詳細な加減速の数値よりも，領域間の遷移（接近）を捉え

ることの方が重要である．したがって，縦方向の空間を意味的に分割する Far 領域の設定は，シナリオの発見において利点があると考えられる．

7.2.4　グリッド手法との比較

2 つの手法を比較すると，グリッド手法では，同一車線内でのわずかな位置ずれや，シナリオの本質に関わらない微細な速度変動もすべて「状態の変化」として記録される．そのため，生成される状態数が膨大になり，人間が目で見て分析するには情報過多となる場合がある．

これに対し，領域分割手法では，意味のある境界（車線境界や前後関係の境界）を越えない限り，物理的な変動はすべて同一の状態として吸収される．交通シナリオの構造を決定づける本質的な変化のみを抽出し，それ以外の変動をノイズとして捨象することで，高い圧縮率と高い検知精度の両立を実現したと言える．また，領域分割手法により，車両速度や継続時間が異なるデータであっても，探索が可能な数量まで抽象化されるため，異なるシミュレーションログ間の比較分析が容易になるという利点も確認された．

7.3　安全性分析における評価

相対加速度と相対速度の比較　表 6.15 の実験結果について述べる．この実験では，抽象化に用いるパラメータとして「相対加速度」と「相対速度」の 2 種類を使用し，それぞれの安全性の判定結果にどのような差異が生じるかを比較した． その結果，相対速度を用いた場合の方が，FN が少ないという結果が得られた．この要因として，相対加速度のみでは「車両が遠ざかっているか否か」を判別できない点が挙げられる．例えば，相対加速度がゼロ（抽象値：ZERO）の場合，2 台とも等速走行している場合でも,「遠ざかっている（安全)」ケースと「接近している（危険)」ケースの両方が混在しうる．そのため，加速度の基準で安全性を判定する場合，相対加速度が’ZERO’ に抽象化された時点で，危険と判断する必要がある．一方，相対速度を基準等する場合遠ざかっている場合と接近している場合が相対速度における’POS’ と’NEG’ に対応しているため，安全な軌道と判断されるケースが増える．以上のことから，FN を抑制し精度の高い分析を行うためには，相対加速度よりも相対速度を指標として用いることが適切であると言える．

データマッピングにおける安全性分析の健全性と課題　安全性検証において最も重要な要件は，危険な事象を見逃さないことである．実験の結果，等区間，車両周辺領域の手法どちらも，FP が 0 であり，健全性を達成していることが確認された．これにより，原理的には具体空間の膨大なログデータを全数確認せずとも，生

成された抽象軌道を確認するだけで，危険判定に至った経緯を追跡可能であるといえる．

一方で，FN の多さが課題となった．この主たる原因は，距離と速度をそれぞれ独立して抽象化したことで，変数間の相関関係が失われた点にある．表 6.20, 表 6.21 を確認すると分かるように 15 領域に分割したものの方が安全と判断されるケースが増えている．これは far 領域を設定しその領域にいた場合は安全であると定義したためである．この定義はシミュレータ上の速度が必ず 70km/h 以内であると仮定した場合の安全な領域を設定しており，これにより安全性評価の精度が上がった．今回具体空間の安全性の判断に用いた，TTC は，車間距離 x と相対速度 v の比率（x/v）で決定される非線形な値である．しかし，本手法ではこれらを独立した領域にマッピングしているため,「距離は十分にあるが，わずかに接近している（速度が負）」といった安全なケースであっても，抽象空間上では「接近中かつ前方領域」という情報しか残らない．このとき，健全性を維持するためには，不確実な領域をすべて「危険」とみなす近似 (Over approximation) を行わざるを得ず，結果として多くの安全なシナリオを危険と判定することになる．

この課題を解決するためには，変数間の関係性を直接保持できる「述語抽象化（Predicate Abstraction）」の導入が有効である．具体的には，距離と速度を独立して扱うのではなく，安全性判定に必要な関係式そのものを抽象化の基準とする．例えば，TTC < 2.0 に相当する述語 $P \equiv (x - 2.0v < 0)$ を定義し，この述語が真か偽かという情報を抽象状態に付加することで，物理量の相関関係を直接的に抽象空間へ持ち込むことが可能となる．これにより，過剰検知を大幅に抑制しつつ，本手法の利点である健全性と可読性を高い次元で両立できると期待される．

したがって，現状の手法はスクリーニングとしての健全性は満たしているものの,「本当に危険なシナリオのみをピンポイントで特定する」高精度な安全性評価を実現するためには，述語抽象化による改良が必要不可欠である．

7.4 提案手法の利点と限界点の考察

7.4.1 健全性が保証されていることの利点

定量的な評価において，車両の周辺位置に基づいた手法はグリッド手法と比較して約 99%という極めて高いデータ圧縮率を達成した．一般に，情報の圧縮率は再現性（精度）とトレードオフの関係にあるが，車両の周辺位置に基づいた手法はカットイン検知において，高い圧縮率を維持しつつ FP ゼロを達成している．よって，この手法は，健全性を保ったまま，必要な情報を保持した抽象化として機能しているといえる．

7.4.2　横方向運動と縦方向運動の抽象化特性

実験結果において，カットインやカットアウト（横方向運動）に関しては，グリッドサイズや領域定義に関わらず高い精度が維持された．これは，カットイン等の事象が「車線境界を跨ぐ」という離散的なイベントであり，本手法で定義した領域境界線が物理空間における車線境界と一致しているためである．すなわち，提案手法による抽象化と対象シナリオの親和性が極めて高い．

一方，加速・減速シナリオ（縦方向運動）に関しては，グリッド手法と比較して領域を分割した手法は，検知の粒度が粗くなる傾向が見られた．グリッド手法では距離が縮まる過程を細かく遷移として捉えられるが，領域分割手法では「Leadの領域内で減速が起こっている」か,「far-front の領域で減速が起こっている」といった限られた状況のみが記録される．

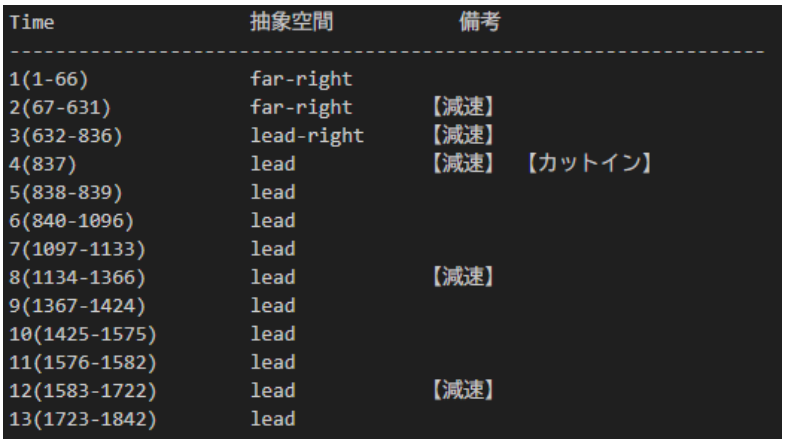
しかし，本研究の目的である目で見て分析できるという観点に立ち返ると，加減速の細かい遷移を厳密に追跡する必要性は低い．JAMA フレームワーク等のシナリオ定義においても，重要なのは「車両同士が危険な距離まで接近したか」という事実である．車両から十分に離れた安全な場所での加減速を検知する必要性は低く，注意すべき領域でどのような動きがあったかを確実に捉えることができれば十分である．その視点で考えると，本手法はリスクが高まるような状態の変化を的確に抽出できているため，抽象化手法として適切であると考える．

7.4.3　複合的なシナリオに対する検出特性の課題

蛇行（Swerve）や U ターンのような複合的なシナリオを検知するためには，原子的な事象（カットイン等）の組み合わせだけで無く，車線周辺領域の定義拡張が必要である．本手法の特性をより詳細に検証するため，複合的シナリオとして，6.3 節で行った実験で得られた Swerve（蛇行）シナリオに対する適用結果を図 7.1 に，U ターンシナリオに対する適用結果を図 7.2 に示す．

Time	抽象空間	備考
1(1-50)	far-right	
2(51-687)	far-right	【減速】
3(688-809)	lead-right	【減速】
4(810-841)	right	
5(842-1195)	follow-right	
6(1196-1389)	far-rear-right	
7(1390-2097)	far-rear	
8(2098-2640)	far-rear-left	
9(2641-2724)	far-rear-left	

図 7.1: 蛇行シナリオの抽象軌道の例



Time	抽象空間	備考	
1(1-66)	far-right		
2(67-631)	far-right	【減速】	
3(632-836)	lead-right	【減速】	
4(837)	lead	【減速】	【カットイン】
5(838-839)	lead		
6(840-1096)	lead		
7(1097-1133)	lead		
8(1134-1366)	lead	【減速】	
9(1367-1424)	lead		
10(1425-1575)	lead		
11(1576-1582)	lead		
12(1583-1722)	lead	【減速】	
13(1723-1842)	lead		

図 7.2: U ターンシナリオの抽象軌道の例

図 7.1 に示す通り，本手法の領域は車両幅に基づいたものであるため，蛇行シナリオでは減速のみしかとらえることができていない．したがって，この抽象軌道から蛇行シナリオであること判断することは難しい．抽象空間の初期位置が far-right から, 最終的に far-rear-left に移動していることとから，おそらく他車が対向車線から自車に向かって走行していることは予測できても，蛇行が発生しているのかについて抽象軌道だけでは判断できない．車両の周辺の領域を使用して蛇行シナリオを検知するには，自車の車幅には侵入していないが本来とるべき距離より近い場合等を検知できるような，周辺の領域を別で設定することが必要であると考えられる．別で設定した領域内では，限定的に領域を細かく分割し，例えば，左右に 2 回以上遷移したら蛇行しているというような演算子を定義することができれば，蛇行シナリオを検知することができると考えられる．

図 7.2 に示す U ターンシナリオの結果を確認すると，抽象空間上では「前方からのカットイン」のように見える結果となっており，両者の区別がつかない結果となった．これは，本手法が車両の「位置」のみを抽象化しており，車両の「向き」や「移動の軌跡形状」に関する情報が含まれていないためである．U ターンとカットインを明確に区別するためには，抽象状態の定義を拡張し，車両の方位角や，領域進入時の角度情報を付加する方法が有効であると考えられる．

7.5 既存手法との比較

7.5.1 lanelet2 と STL を用いた手法との比較

Lanelet2 を用いた手法は，グリッドによる離散化ではなく，車線の中心線に沿った位置関係を用いて前後関係を取得する．そのため，グリッド幅に依存する本手法と比較して，より高精度な前後位置の特定が可能であるという利点を持つ．

表 6.24 および表 6.25 の結果が示すように、マップ設定（Road width）が実際の車線幅（約 3.5 m）と整合している範囲（1.9 m〜4.0 m）では、TP が高く FN が 0 でありこの範囲であれば健全性が保たれ，理想的な検出性能を示している．しかし、設定値を 7.0 m や 8.0 m のように実際の車線幅から乖離させた場合、検出精度が著しく低下（FN が増加）する傾向が見られた．これは、STL による判定が「車線境界までの絶対的な距離」などの幾何学的な閾値に強く依存しているためである．したがって、地図データへの依存度が高い既存手法は、環境変化や地図誤差に対して脆弱であるといえる．

対して、本研究の提案手法は「自車を中心とした相対的な領域」を用いているため、詳細な高精度地図や厳密な車線幅定義を必要としない．これにより、地図データが整備されていない環境においても、柔軟にシナリオを抽出できるという点で優位性があると考えられる．なお、加速・減速シナリオに関して FP が発生している要因については、STL の定義が「速度変化」を厳密に追跡しているのに対し、今回の比較実験では位置変化（車間距離の詰まり）も含めた広義の定義との間で解釈の齟齬が生じたためと考えられる．

lanelet と STL を用いた手法のデメリットとしては，定義したシナリオに適さない場合，検出することができない点にある．STL を用いた手法は，あらかじめ定義された論理式に対する真偽値（True/False）を判定するアプローチであるため，事前に定義されていない未知のシナリオや想定外の挙動を検出・記述することは困難である．一方，提案手法は軌道を抽象的なシンボル列（文字列）に変換するなアプローチである．そのため，事前に定義していない挙動であっても，「どのような領域遷移を経て移動したか」という構造化された情報を保持しており，事後的な分析や類似検索が可能である．この点において，提案手法は未知の状況に対する汎用性が高いといえる．

7.5.2 LLM を用いた手法との比較

LLM を用いた実験の結果，LM は個々の車両挙動に対する定性的な説明能力においては優れていることが確認された．特に，秒数を用いて状況がどれくらい続いているかといった説明はほかの手法では行われていないため，このような情報を知りたい場合には有用であると考えられる．しかし、定量的なシナリオ抽出（JAMA フレームワークへの対応）という観点では，以下の 3 つの課題が明らかとなった．

一つ目は，出力がプロンプト設計に依存する点である．「軌道の状態について説明してください」といった一般的な指示を与えた場合，LLM は自車の挙動（速度変化や加減速）の記述に終始する傾向が見られた．例えば，前方に他車が割り込んで減速したケースであっても，「自車が減速した」という事実のみが記述され，「他車の割り込み（カットイン）に起因する」という相互作用の文脈が無視される場合がある．JAMA フレームワークにおける交通外乱シナリオは，自車と他車の位置関係および相互作用によって定義されるため，自車の動き単体の説明では不十

分である．これを回避するためには，入力データの構造や指示を厳密に設計する必要があり，汎用的な利用の障壁となる．したがって，JAMA フレームワークのようなシナリオにおいては，自車だけでなく他車との位置関係が重要となるため，自車の動きが適切に説明されているだけでは不十分である．また,「プロンプトでカットインしていますか」と聞いた場合カットインしていると答えるが,「状況を説明してください」とするとカットインとは答えない．このことから，LLM が使用者の意図を汲んで迎合する可能性が示唆される．原因として，LLM 側でカットインするという基準を持っているのではなく，カットインしているかと聞かれたため，横方向の移動をより厳密に調べようとして，カットインが検知できているのではないかと考えられる．

二つ目は，レーン情報の欠如による認識の限界である．プロンプトで「この軌道は 2 台の車両の軌道である」と明記し，相対位置関係を認識させた場合であっても，LLM による説明は不十分な場合がある．LLM は与えられた座標データの数値的な変化に基づき,「南東方向への移動」や「接近」といった幾何学的な挙動を説明することは可能である．しかし,「カットイン」という事象は車線や，車幅など境界をまたぐことによって定義される．軌道データ（数値列）のみを入力とする LLM には，この「車線」という環境情報が与えられていないため，単なる斜行と車線変更を明確に区別して判定することが原理的に困難である．しかし，絶対値の位置関係で判断をするため，南東方向等，俯瞰した動きの説明はすることができるが，レーン情報などを考慮して，カットインしていることまでは判断ができない．

三つ目は，大量のデータを処理における実用性の課題である．実際にシミュレータを用いて軌道を処理する場合，数千〜数万件のファイルを処理必要がある．しかし LLM ではコンテキストウィンドウの制限により，大量のファイルを一括処理することは困難である．また，入力データ量が増加すると，ハルシネーションが発生する確率も高まる．そのため，膨大な全軌道データを LLM に投入して全数検査を行うことは，時間的・コスト的・信頼性の観点から現実的ではない．

以上のことから，LLM は抽出された特定シナリオより詳しく見たい場合には適しているものの，JAMA フレームワークに基づくシナリオを大量のログから機械的に抽出・スクリーニングするタスクにおいては，本研究の提案手法のような軽量かつ定義が明確なアルゴリズムの方が適していると考えられる．

7.5.3　提案手法と既存手法の特性比較

これまでの比較実験および考察の結果を総括し，領域分割手法，均一グリッド法，Lanelet2/STL 法，および LLM との特性比較を表 7.1 に示す．各評価項目の詳細な比較考察は以下の通りである．

抽象化の質と可読性　LLM は自然言語処理能力に長けており,「減速（Deceleration）」や「割り込み（cut-in）」といった提案手法で定義した状況以外にも,「強い減速」「スムーズに接近」など状況が想像しやすい説明をするため可読性は最も高いといえる（◎）．提案手法においては，事前に定義した状況に限られるが,「カットイン」などの状況を説明することができる．グリッドと，領域を分割した手法を比較すると，生成される抽象軌道において，相対位置を数値の羅列となる均一グリッド法（△）よりも領域を分割したものの方が，直感的に理解しやすい軌道であるといえる（○）．また，この抽象軌道は構造化されたテキストデータであるため，データベースに蓄積することで，SQL 等のクエリを用いて特定のシナリオを高速に検索・抽出できるという利点も有する．なお，STL は論理式としての厳密性は高いものの，与えられた軌道がシナリオ条件を満たすか否か（True/False）を判定する手法であり，軌道の挙動そのものを記述するわけではないため，可読性の観点では評価対象外 (-) とした．

ロバスト性と地図依存性　STL を用いた手法は，地図データ（車線幅など）が正確である場合には極めて高い精度（◎）を発揮するが，実験で示した通り，地図情報と実環境に乖離がある場合には検出漏れを起こすリスクがある．対して提案手法は，自車を中心とした相対的な位置関係のみに依存するため，高精度地図を必要としない（地図データ非依存）．これにより，地図更新の遅れや白線の摩耗といった環境要因の影響を受けにくく，実環境におけるロバスト性は極めて高い（◎）と言える．

健全性とハルシネーション耐性　領域分割手法，均一グリッド手法，および STL を用いた手法は，幾何学的な配置をマッピングするため，ハルシネーションが発生する余地がなく，健全性が保証される（◎）．一方，LLM は確率的な出力を行うため，存在しない座標を出力するハルシネーションのリスクが避けられず，安全性保証の観点からは信頼性に課題が残る（△）．

汎用性と記述能力　STL は「事前に定義した論理式」に対する真偽判定を行う「判別的アプローチ」であるため，想定外のシナリオや未定義の挙動に対しては無力であり，汎用性は限定的である（△）．これに対し，提案手法はあらゆる軌道を状態遷移列として出力する「記述的アプローチ」である．未知の挙動であっても「どのような遷移を辿ったか」という形式で記録・分類が可能であるため，探索的な分析において高い汎用性を発揮する（◎）．

計算コストと実用性　シミュレーションによる大規模検証では，数万件規模のデータを処理する必要がある．LLM は 1 件あたりの計算コストが高く，コンテキストウィンドウの制限等もあるため全数検査には不向きである（△）．提案手法は軽量

な幾何計算のみで動作するため計算コストが非常に低く，大規模データセットに対するスクリーニング技術として極めて高い実用性を持つ（◎）.

表 7.1: 提案手法と他手法の特性比較

評価項目	領域分割手法	均一グリッド法	Lanelet / STL	LLM (Generative AI)
抽象化の基準	車両周辺領域	グリッド	論理式	自然言語・文脈
可読性	○	△	-	◎
地図データ非依存	不要	不要	必要	不要
ロバスト性	◎	○	◎	△
健全性	◎	◎	◎	△
汎用性	◎	◎	△	◎
計算コスト効率	◎	◎	◎	△
再現性	◎	◎	◎	△
シナリオの識別精度	○	○	◎	△
ハルシネーション耐性	◎	◎	◎	△

評価基準：◎（優れている/低い），○（普通/条件付き），△（劣っている/高い）

※計算コストは「低い」場合を優れている（◎）と判定。

第8章　おわりに

本研究では，自動運転システムの安全性検証において課題となっている、シミュレータの低レベルなログデータと JAMA フレームワークの高レベルなシナリオ記述との間にある「抽象度のギャップ」に着目した．この課題を解決するため，等間隔のグリッドを用いたデータマッピングと自車周辺の領域を分割したデータマッピングによる軌道抽象化手法を提案し，シミュレーションデータを用いた評価実験を行った．

実験および考察の結果，得られた知見は以下の通りである．

一つ目は，車両周辺の領域を分割した手法は高いデータ圧縮率と可読性の両立を実現することができた点である．等間隔のグリッドを用いた場合で明らかになったように，等間隔のグリッド手法では「シナリオの識別精度」と「可読性」の間にトレードオフが存在し，安定的に人間が解釈可能なレベルまで状態数を削減することが困難であった．対して，車両周辺の領域を分割した手法は JAMA フレームワークの定義と整合するような境界を用いることで，約 99 %のデータ圧縮を達成した．これにより，数万ステップの数値の羅列を「人間が直感的に理解可能な状態遷移」への変換することに成功した．これにより，専門家でなくとも車両の挙動を一目で把握可能となり，大規模なログデータの分析効率を大幅に向上させることを示した．

二つ目は，安全性検証における健全性の実証である． 実験において，FN は 0 件であり，危険なシナリオを見逃さなかったといえる．評価実験の結果，提案手法は全ての危険シナリオに対して見逃しゼロ（FN=0）を達成した． これは，提案手法が採用した抽象化が，危険な事象を漏らさない健全な抽象化として機能していることを示している．これにより，具体空間の膨大なログデータを全数検査せずとも，生成された抽象軌道を確認するだけで危険判定が可能となり，検証プロセスの効率化と信頼性向上に寄与することを確認した．

三つ目は，開発プロセス全体への貢献である． 生成された抽象軌道は検索可能なテキストデータとなるため，特定のシーン定義に合致するログの高速な抽出や，類似シナリオのクラスタリングに応用することもできる．これは，事故に至らないまでも危険な挙動（ヒヤリハット）の定性的なモニタリングや，未知の危険シナリオ発見に対する強力な支援ツールとなることが示唆された．

一方で，解決すべき課題も明らかとなった．一つ目は，安全性判定における FP の抑制である．現状では，距離と速度を独立して抽象化しているため，衝突リスク（TTC）の推定において FP が発生する傾向にある．この解決策として，変数

間の相関関係を直接保持できる「述語抽象化」の導入が有効であると考えられる．

二つ目は，蛇行シナリオのような複合的なシナリオへの対応である．現状の原子的なイベント（カットイン単体など）の検出に加え，それらが時系列でどのように組み合わさったかを判定する上位のロジック構築や，車線周辺領域の定義拡張が必要となる．

本研究で提案した抽象化手法は，自動運転の安全性を効率的かつ網羅的に説明するための基盤技術として，今後の自動運転開発に貢献できるものと考える．

謝辞

本研究を遂行するにあたり，研究計画の立案から論文の執筆に至るまで，多大なるご指導と助言を賜りました，指導教員である北陸先端科学技術大学院大学の青木利晃教授に心より感謝の意を表します．青木先生の熱心なご指導がなければ，本研究を完遂することはできませんでした．

また，本学の冨田尭准教授には，日頃の研究報告等において，多角的な視点から有益なご助言とご鞭撻を賜りました．ここに深く感謝申し上げます．

また，日頃の研究活動において，有益な助言や活発な議論を交わしていただいた青木研究室の皆様に深く感謝いたします．皆様との日々の助言や温かい励ましは，研究を進める上で大きな支えとなりました．

最後に，在学中の生活を支え，常に温かく見守ってくれた家族に深く感謝します．

参考文献

[1] Walther Wachenfeld and Hermann Winner. The release of autonomous vehicles. *Autonomous Driving: Technical, Legal and Social Aspects*, pp. 425–449, 1 2016.

[2] Till Menzel, Gerrit Bagschik, Leon Isensee, Andre Schomburg, and Markus Maurer. From functional to logical scenarios: Detailing a keyword-based scenario description for execution in a simulation environment. *IEEE Intelligent Vehicles Symposium, Proceedings*, Vol. 2019-June, pp. 2383–2390, 6 2019.

[3] Daniel J. Fremont, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Scenic: A language for scenario specification and scene generation. *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pp. 63–78, 6 2019.

[4] Siddartha Khastgir, Simon Brewerton, John Thomas, and Paul Jennings. Systems approach to creating test scenarios for automated driving systems. *Reliability Engineering & System Safety*, Vol. 215, p. 107610, 11 2021.

[5] Briti Gangopadhyay, Siddartha Khastgir, Sumanta Dey, Pallab Dasgupta, Giovanni Montana, and Paul Jennings. Identification of test cases for automated driving systems using bayesian optimization. *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, pp. 1961–1967, 10 2019.

[6] 一般社団法人日本自動車工業会. 自動運転の安全性評価フレームワーク ver 3.0. 2022.

[7] Sebastian Geyer, Marcel Baltzer, Benjamin Franz, Stephan Hakuli, Michaela Kauer, Martin Kienle, Sonja Meier, Thomas Weigerber, Klaus Bengler, Ralph Bruder, Frank Flemisch, and Hermann Winner. Concept and development of a unified ontology for generating test and use-case catalogues for assisted and automated vehicle guidance. *IET Intelligent Transport Systems*, Vol. 8, pp. 183–189, 5 2014.

[8] Till Menzel, Gerrit Bagschik, , and Markus Maurer. Scenarios for development, test and validation of automated vehicles. *IEEE Intelligent Vehicles Symposium, Proceedings*, Vol. 2018-June, pp. 1821–1827, 10 2018.

[9] Barbara Schütt, Thilo Braun, Stefan Otten, and Eric Sax. Sceml: A graphical modeling framework for scenario-based testing of autonomous vehicles. *Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2020*, pp. 114–120, 10 2020.

[10] Lu Huang, Huawei Liang, Biao Yu, Bichun Li, and Hui Zhu. Ontology-based driving scene modeling, situation assessment and decision making for autonomous vehicles. *2019 4th Asia-Pacific Conference on Intelligent Robot Systems, ACIRS 2019*, pp. 57–62, 7 2019.

[11] L. Kovacs and O. Semerath. Towards the formal semantics of scenario tests for autonomous vehicles. In *2021 10th Latin American Symposium on Dependable Computing Ladc 2021 Proceedings*, 2021.

[12] D.J. Fremont, E. Kim, Y.V. Pant, S.A. Seshia, A. Acharya, X. Bruso, P. Wells, S. Lemke, Q. Lu, and S. Mehta. Formal scenario-based testing of autonomous vehicles: From simulation to the real world. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems ITSC 2020*, 2020.

[13] Lars Klitzke, Carsten Koch, Andreas Haja, and Frank Köster. Real-world test drive vehicle data management system for validation of automated driving systems. *VEHITS 2019 - Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems*, pp. 171–180, 2019.

[14] Ma Qianli, Wang Tong, Chen Jinglei, and Chen Hongyu. Connected automated vehicles and its, vehicle dynamics and control data-driven identification and analysis of road test scenarios for self-driving minibuses. *International Journal of Automotive Technology*, 123.

[15] Minhee Kang, Saeyan Eom, and Keeyeon Hwang. Advancing autonomous vehicle safety assessment: A novel methodology for moving from functional to concrete scenarios using kinetic 3d-lidar and shap. *Results in Engineering*, Vol. 24, , 12 2024.

[16] Minhee Kang, Jungwook Seo, Keeyeon Hwang, and Young Yoon. Critical voxel learning with vision transformer and derivation of logical av safety assessment scenarios. *Accident Analysis and Prevention*, Vol. 195, , 2 2024.

[17] Jesse Reimann, Nico Mansion, James Haydon, Benjamin Bray, Agnishom Chattopadhyay, Sota Sato, Masaki Waga, Étienne André, Ichiro Hasuo, Naoki Ueda, and Yosuke Yokoyama. Temporal logic formalisation of iso 34502 critical scenarios: Modular construction with the rss safety distance. *Proceedings of the ACM Symposium on Applied Computing*, pp. 186–195, 4 2024.

[18] H. Muslim, S. Endo, H. Imanaga, S. Kitajima, N. Uchida, E. Kitahara, K. Ozawa, H. Sato, and H. Nakamura. Cut-out scenario generation with reasonability foreseeable parameter range from real highway dataset for autonomous vehicle assessment. *IEEE Access*, Vol. 11, pp. 45349–45363, 2023.

[19] Patrick Cousot and Radhia Cousot. Abstract interpretation: "a" unified lattice model for static analysis of programs by construction or approximation of fixpoints. *Conference Record of the Annual ACM Symposium on Principles of Programming Languages*, Vol. Part F130756, pp. 238–252, 1 1977.

[20] Meng Jun Li, Zhou Jun Li, and Huo Wang Chen. Program verification techniques based on the abstract interpretation theory. *Ruan Jian Xue Bao/Journal of Software*, Vol. 19, pp. 17–26, 1 2008.

[21] Patrick Cousot. Formal verification by abstract interpretation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 7226 LNCS, pp. 3–7, 2012.

[22] Jan Reineke and Reinhard Wilhelm. Static timing analysis – what is special? *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 9560, pp. 74–87, 2016.

[23] Manuel V. Hermenegildo, Germán Puebla, Francisco Bueno, and Pedro López-García. Integrated program debugging, verification, and optimization using abstract interpretation (and the ciao system preprocessor). *Science of Computer Programming*, Vol. 58, pp. 115–140, 10 2005.

[24] Bor Yuh Evan Chang and K. Rustan M. Leino. Abstract interpretation with alien expressions and heap structures. *Lecture Notes in Computer Science*, Vol. 3385, pp. 147–163, 2005.

[25] Olivier Ponsini, Claude Michel, and Michel Rueher. Verifying floating-point programs with constraint programming and abstract interpretation techniques. *Automated Software Engineering*, Vol. 23, pp. 191–217, 6 2016.

[26] 田辺良則, 高井利憲, 高橋孝一. 抽象化を用いた検証ツール. コンピュータ ソフトウェア, Vol. 22, pp. 2–44, 1 2005.

[27] The autoware foundation. https://github.com/autowarefoundation.

[28] Tier IV, Inc. 第 5 回：oss 自動運転ソフトウェア「autoware」の正体, 2024. https://tier4.jp/media/detail/?sys_id=8etDl805QwfbfjBdcLVsy&category=BLOG (参照 2026-01-28).

[29] tier4/awsim: Open sourced digital twin simulator for autoware. https://github.com/tier4/AWSIM.

[30] D. N. Lee. A theory of visual control of braking based on information about time to collision. *Perception*, Vol. 5, pp. 437–459, 1976.

[31] fomaad/aw-runtime-verification. https://github.com/fomaad/AW-Runtime-Verification/tree/main/Traces.

[32] fomaad/simulation-data: Contain various trace data collected through running autoware. https://github.com/fomaad/Simulation-Data.