MathWorks®

ランタイムエラーが存在しないコードの証明

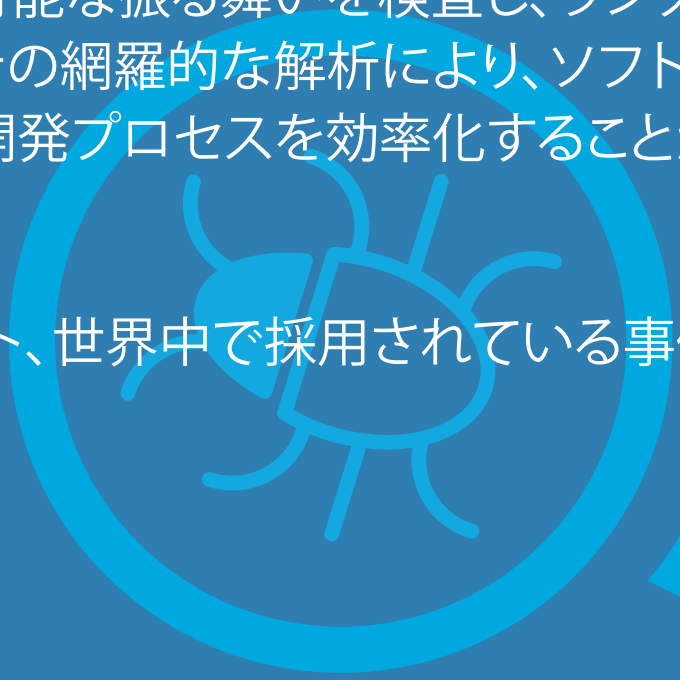
形式検証を用いた静的解析 4つのメリット

次世代の組み込みシステムに求められる検証技術

製品の高機能化に伴うシステムの複雑化、ソフトウェアの大規模化などに対応するため、世界中の先進的な組み込みシステムの開発現場では、網羅的・効率的な検証手法の採用が進んでいます。

実績豊富な形式手法である抽象解釈を用いた静的検証は、一般的な静的解析技術と、動的検証のギャップを埋めます。抽象解釈は、プログラムを実行せずに、プログラム内ですべての可能な振る舞いを検査し、ランタイムエラーの有無を数学的に証明します。その網羅的な解析により、ソフトウェアの高い信頼性を確保すると同時に、開発プロセスを効率化することができます。

抽象解釈を用いた静的解析のメリット、世界中で採用されている事例をご覧ください。



ソフトウェアテストの手法

手法	概要	特徴
コードレビュー	ソースコードを人の目でレビューする作業です。 小規模なアプリケーションではバグの発見やコードの改善に有効です。	コード規模と複雑性の増大に伴い、テスト工数や見落としが増えるため、網羅的なコードレビューは非常に困難です。
動的解析	テストケースを用意し、ソフトウェアを実行して検証する作業です。動作確認として必要ですが、有用性はテストケースに依存します。	テストケースは無限に存在するため網羅的な検証はできません。
一般的な静的解析	ソフトウェアを実行せずに欠陥検出を行う作業です。欠陥・エラーの検出や、コーディング規約への準拠、コードメトリクスの測定に有効です。	動的な挙動によるエラーを発見できません。
形式検証を用いた静的解析	ソフトウェアを実行せずに動的な特性を検証します。網羅的な検証により欠陥やエラー、セキュリティ脆弱性を発見します。 また、コーディング規約への準拠やコードメトリクスの測定も行います。	ランタイムエラーが存在しないことを証明できるため、高い信頼性を担保します。 動的解析に必要なテストケースを作成・実行するコストが不要であり、修正が最も容易な開発の初期段階でバグを発見・修正できるため、開発プロセスを効率化します。

抽象解釈でコードの正しさを証明するメリット

1. コードの信頼性の確保

網羅的なコード検証で抽象解釈はランタイムエラーを見つけ、さらには、コードの正しさを証明します。 この「証明」は、システム故障が許されないミッションクリティカルなアプリケーションには極めて重要です。

通常のデバッグツールではエラーを発見することができても、コード全体の信頼性は証明しません。

抽象解釈を用いると、コードがシステム故障を引き起こさないことを証明できるので、ソフトウェアの高い信頼性を実現できます。

"Polyspace 製品によって、高レベルなソフトウェアの信頼性を確保することができます。これは、業界内の他のツールにはできないことです。"

— 菊池光彦氏, 日産自動車

"Polyspace Code Proverによって非常に高い信頼性を確保できるので、ランタイムエラーがないと確信を持ち、コードレビューや検証を加速することができます。"

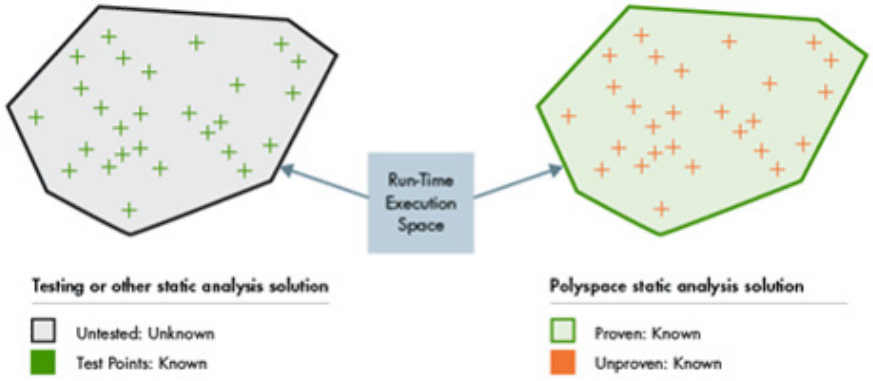
— Lars Schiemanck, Miracor

抽象解釈でコードの正しさを証明するメリット

1. コードの信頼性の確保

抽象解釈とは実績豊富な形式手法のひとつです。この技法はソフトウェアコンパイル時の動的性質を検証し、一般の静的解析技術と動的検証のギャップを埋めます。

抽象解釈はプログラムを実行せずに、プログラム内で全ての可能な振る舞い—全ての入力の組み合わせと実行シーケンス—を検査して、ランタイムエラーが発生する状況を明確にします。



"ハンドコード内のランタイムエラーを100%特定、削除することができました。"

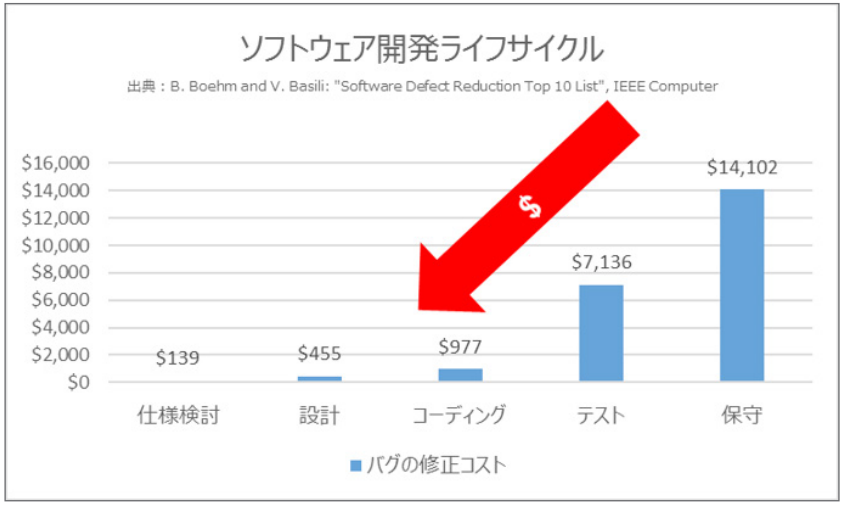
— Jungho Moon, 大韓航空

抽象解釈でコードの正しさを証明するメリット

2. 効率の向上

アプリケーションの動的性質を検証することで、 抽象解釈はランタイムエラーが存在しないコードと信頼性の侵害に繋がるコードを識別できます。

コードを実行する前にエラーを指摘することで、修正が最も容易な段階でエラーを消去して**時間とコストを大きく削減します。**



"Polyspace 製品は、どのコードを実行した時にランタイムエラーが発生しうるかを明らかにするだけでなく、実行条件に関係なく決してランタイムエラーが発生しないコードを特定することができます。しかもコーディング中、つまり単体テストの前に検証できるため、サプライヤーにとって非常に大きな価値があります"

— 菊池光彦氏, 日産自動車

"グリーン：正常と判定されたコードにはランタイムエラーがないと証明されているので、その後のコードレビューの段階では、オレンジ：未証明のコードだけに集中することができました。"

— Lars Schiemanck, Miracor Medical Systems

抽象解釈でコードの正しさを証明するメリット

3. オーバーヘッドの削減

抽象解釈はコードを実行する必要がないため、テストケースを作成・実行するコストなく高精度の解析結果を得ることができます。

また、抽象解釈は進行中の開発プロセスにも、手を加えることなく実行可能です

"フライト中に起こりうる危険なエラーを排し、安全なソフトウェアの設計に、Polyspace製品は必須です。

Polyspaceでハンドコード中のゼロ除算、オーバーフローを数多く見つけ、また、Embedded Coderで生成したコードにランタイムエラーがないことを証明しました。"

— Jungho Moon, 大韓航空

"Polyspaceを使うとテストケースを書かずにバグを検出することができます。

また、コードレビューで見落とされた重大なバグを見つけることができました。"

— Ralph Paul, Solar Impulse

© Solar Impulse | Revillard | Rezo.ch

抽象解釈でコードの正しさを証明するメリット

4. デバッグ作業の容易化

抽象解釈は、エラーの症状だけではなく、エラーの発生点を指摘するため、**デバッグを容易化します。**

エラーの原因を分析するためのトレース作業や、再現するための時間を削除することができます。

抽象解釈は徹底的な解析の上、リピート可能です。コード内の各操作は自動的に全ての入力の組み合わせを取り組んで指摘、解析、検査します。

"Polyspace 製品によって、従来のテストよりはるかに低いコストで、アプリケーションが確実に動作することが担保されます。機能テストがランタイムエラーによって妨げられなくなったことは、言うまでもありません。"

— 菊池光彦氏, 日産自動車

"Polyspaceによって、エラーのないコードと詳細確認が必要なコードが識別されるため、的を絞ってデバッグを行うことができました。"

— Dr. Karen Gundy-Burlet, NASAエイムズ研究センター

抽象解釈について

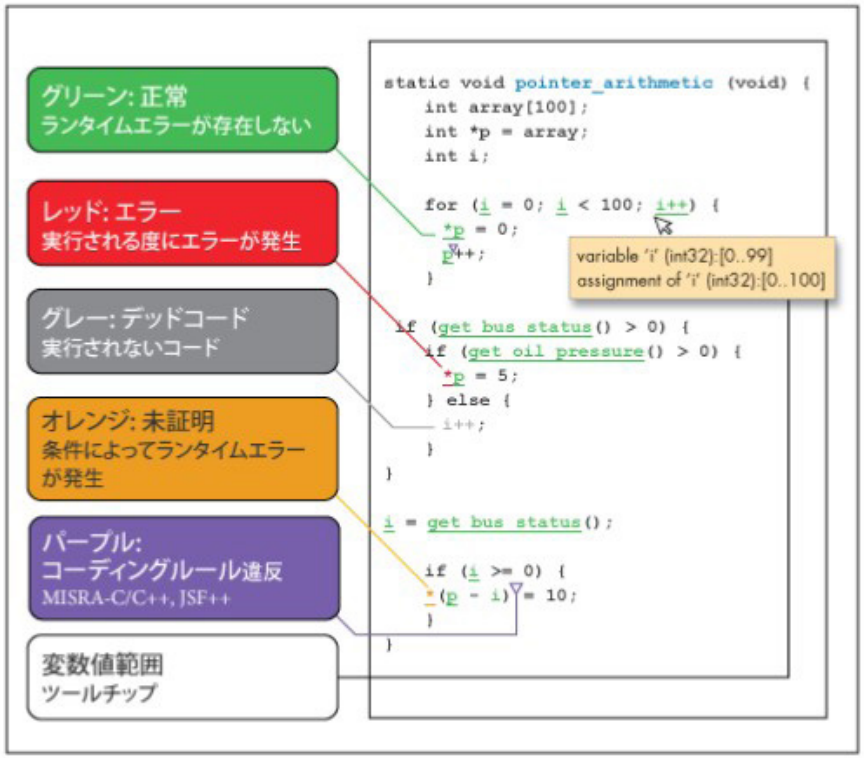
抽象解釈はソフトウェアアプリケーション等の複雑な動的システムに対してルールを定義する幅広い数学的な定理を使用して解析します。

プログラムの各状態を一般化して表して、操作させるルールを定義します。抽象解釈は数学的な抽象化だけではなく、抽象化を解釈します。

プログラム状態を数学的に抽象化するため、コード内の全ての変数を徹底的に分析します。以前はこの解析に対応できる処理能力を容易に入手することはできませんでした。現在のアルゴリズムと処理能力の増加により、複雑な検証の課題に対して抽象解釈は現実的な解決案です。

ランタイムエラー検出に適用することで、抽象解釈は全ての危険性がある操作を検証し、各操作が正常、エラー、非到達か、未証明と自動的に判断します。エンジニアはテストの最も早期段階であるコンパイル時に抽象解釈を使用できます。

Polyspace Code Proverの解析結果は各操作を色付けしてランタイムエラーの存在・可能性・不在とデッドコードを明確に表示します。



Polysoace Code Proverによるランタイムエラー証明

ユーザー事例

日産自動車、ソフトウェアの信頼性を向上

課題

ソフトウェア品質を向上させるために、発見が困難なランタイムエラーを特定する

ソリューション

MathWorks のPolyspace製品を使用して、日産とサプライヤーのコードを包括的に解析する

結果

- サプライヤーのバグを検出して評価
- ソフトウェアの信頼性が向上
- 日産のサプライヤーが Polyspace 製品を採用。

Miracorが高度管理医療機器（クラスIII）ソフトウェアのラインタイムエラーを根絶しテスト期間を短縮

課題

高度管理医療機器（クラスIII）の安全性を担保し、

ステント結果を改善する

ソリューション

Polyspace Code Proverでランタイムエラーがないことを証明し、コードレビューを効率化し、機能テストを補完し、 認証取得のための検証プロセスを作り上げる

結果

- 欠陥のあるコードや使用されていないコードを特定
- 認証取得のための検証プロセスの確立
- コードレビューの効率化

ユーザー事例

大韓航空がモデルベースデザインでUAVのフライトコントロールソフトウェアの開発と検証を高速化

課題
無人航空機用フライトコントロールソフトウェアの開発と検証

ソリューション
モデルベースデザインでフライトコントロールモード、運転ロジックの設計とシミュレーションを行い、量産コードの生成と検証、HILテストを実施

結果

- ハンドコード内のランタイムエラーを100%特定、削除
- 開発工数を60%削減
- 高価な実機飛行テストを最小化

Solar Impulseが先進的な太陽光発電航空機を開発

課題
地球一周が可能な太陽光発電航空機の開発

ソリューション
MATLAB/Simulinkでシステムモデルを作成して設計トレードオフを分析し、航空力学モデルを作成して高精度のシミュレーションとパイロットトレーニングを実施

結果

- 主要な設計を早期に決定
- 重要なパイロットトレーニングが実現
- モデルは開発全体を通じて共有、再利用

© Solar Impulse | Revillard | Rezo.ch

ユーザー事例

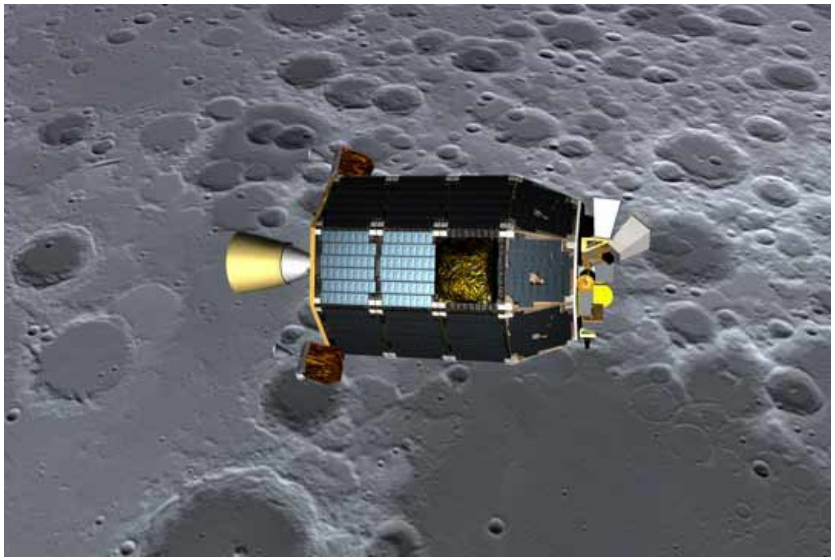
NASAエイムズ研究センターが
月大気・塵探査機LADEE用の飛行ソフトウェアを開発

課題
宇宙探査機LADEE用搭載用の飛行ソフトウェアの開発

ソリューション
モデルベースデザインを採用し、制御システムと宇宙探査機LADEEをモデル化、26,000行のCコードを生成、HILとPILテストを実施し、ミッショントレーニングシミュレーターを作成

結果

- トレーニングと指令の検証にモデルを再利用
- 飛行ソフトウェアを軌道上でシームレスに更新
- 正式なコード検査のプロセス合理化



参考資料

Polyspaceによる静的解析について、詳細は以下をご参照ください。

観る

次世代システムに求められるソフトウェア検証技術 ～静的解析の価値と有効性～ (28:57)

読む

Polyspace製品を使用した包括的な静的解析

ランタイムエラーが存在しないコードを証明するPolyspace

93110v00 11/19