


令和6年度電子情報工学科 卒業研究中間報告					
報告者	06席	氏名	糸川 倫太郎	研究室	青山研究室
論文題目	A JavaScript compiler and TypeScript checker with a focus on static analysis and runtime performance				
キーワード	JavaScript, TypeScript, compiler, static analysis				

＜背景＞AltJS として寡占的な立ち位置を獲得した TypeScript という言語がある．その根幹となる TypeScript を JavaScript に変換するコンパイラ（tsc）は，静的解析機や language server に使用されてきた．しかし，そんな tsc にはある問題がある．それは，tsc による TypeScript の解析と型推論は非常に計算コストと時間コストが高いことである．これにはいくつかの解決策があるが，どれも tsc に依存してしまうため，JavaScript ランタイムの起動から実行までのボトルネックを無視できない．そこで，本研究では tsc の代替となるパフォーマンスを意識したコンパイラを作成することにした．

＜目的＞language server や静的解析機で使用した際，tsc と比較して，高速でハイパフォーマンスなコンパイラを作成する．

＜研究の概要＞「decaf」と名付けた本研究は Rust を使用して開発する．TypeScript のパースは oxc_parser を参考に実装する．型推論は tsc を参考に実装する．

decaf では tsc と異なった型推論のアーキテクチャを提案する．



```
graph LR; A[lexing] --> B[parsing]; B --> C[type synthesis & checking]; C --> D[output];
```

図 1: tsc の型推論アーキテクチャ

図 1 は従来の TypeScript の型推論アーキテクチャである．decaf では，このアーキテクチャのうち型の合成と検査の部分が tsc と異なる．

tsc では型の合成と検査が同時に行われるが，decaf ではこれらの処理を分離する．型の合成とは，TypeScript の構文解析木（AST）を decaf の型合成機が解釈できる形（TypeID）に変換することである．decaf では，型検査に他の言語のアイデアを持ち込むために，型の合成した後で，型検査するアーキテクチャを採用した．

decaf では型合成機は，値と構造に関する情報を型として保持し，関数やブロックの振る舞いをイベントとしてエンコードする．型やイベントは，AST の走査中に生成され，型検査機に渡され使用される．

decaf では型検査機は，tsc と異なったアプローチを取る．decaf の型検査機では型を「その値が取りうる可能性の空間」として捉える．これにより，AST を走査しながら型の空間を広げていくことで，実行時に起こりうるすべての可能性を型チェック時に考慮できる．

＜研究の現状＞現在，decaf は TypeScript のほぼ全ての構文をパースできる．一方で型推論，検査は概要で述べたアーキテクチャに基づいて実装中である．

＜今後の方針＞サポートする構文を増やし，型推論の精度を向上させる．また，プレイグラウンドを作成し，フィードバックを受けやすい環境を整える．