Interlinear Layout for Scientific PDFs

과학 PDF를 위한 행간 대역 배치

Dynamic languages such as JavaScript are harder to compile than statically typed ones.
JavaScript와 같은 동적 언어는 정적 타입 언어보다 컴파일하기가 더 어렵다.
Since no concrete type information is available, traditional compilers need to emit generic code. We present a naïve but effective alternative. It works well in practice.
구체적인 타입 정보가 없으므로 전통적 컴파일러는 범용 코드를 생성해야 한다. 우리는 단순하지만 효과적인 대안을 제시한다. 이 대안은 실제로 효과적으로 작동한다.
The first prototype runs on the α-release of the engine and records hot loops that are compiled into efficient machine code without any interpreter overhead.

첫 번째 프로토타입은 엔진의 α-release에서 실행되며, 핫 루프를 기록한 뒤 이를 인터프리터 오버헤드가 없는 효율적인 기계어 코드로 컴파일한다.

Our evaluation covers eleven benchmarks from the SunSpider suite. Speedups range from
평가는 SunSpider 제품군의 11개 벤치마크를 대상으로 한다.
1.2x to 7.5x, with a geometric mean of 3.1x (see Section 4). Compilation time is small
성능 향상은 1.2배에서 7.5배까지이며, 기하평균은 3.1배이다(4절 참조).
compared to execution time, and memory usage grows linearly with the number of traces.

컴파일 시간은 실행 시간에 비해 짧으며, 메모리 사용량은 트레이스 수에 따라 선형적으로 증가한다.

We conclude that trace-based compilation is a practical approach for dynamic languages.
트레이스 기반 컴파일은 동적 언어를 위한 실용적인 접근 방식이라고 결론 내린다.
Future work includes support for recursion and more precise type speculation [12].
향후 과제에는 재귀 지원과 더 정밀한 타입 추측 [12]이 포함된다.