
Dominik Heeb


Fabian Keller

Diplomanden	Dominik Heeb, Fabian Keller
Examinator	Prof. Dr. Luc Bläser
Experte	Prof. Dr. Luc Bläser
Themengebiet	Software

A Study of Dynamic Parallel Checking Methods

Parallel Programming

```
// ... Deklaration a,b,locka,lockb ...

Thread.Start(() =>                  // start thread2 (T2)
{
        lock(lockb) {
                b = 3;
        }
        b = 4;
        lock(locka) {
                Console.WriteLine(a);
        }
});
Thread.Start(() =>                  // start thread3 (T3)
{
        lock(locka) {
                a = 2;
        }
        b = 5;
        lock(lockb) {
                Console.WriteLine(b);
        }
})
lock(locka) {
        a = 3;
}
lock(lockb) {
        b = 6;
}
Console.WriteLine(a);
```

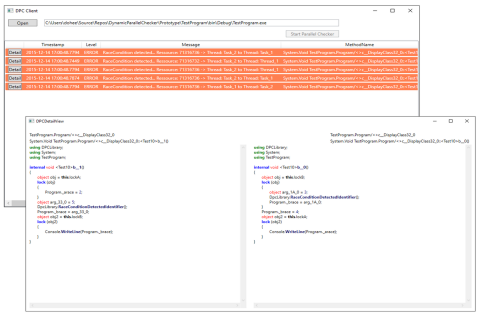
Beispielcode für den Vector Algorithmus

Vektor	Ressource	ThreadNr
(1,3,2)	locka	2
(3,1,4)	lockb	3

ThreadNr	Ressource	Vektor	ZugriffsTyp
2	b	(0,1,0)	write
1	a	(1,0,0)	write
3	a	(1,0,2)	write
...	...	...	...
2	a	(1,3,2)	read
3	b	(3,1,4)	read
1	a	(4,1,0)	read

$$T1 = \begin{pmatrix} T1 & 4 \\ T2 & 1 \end{pmatrix}, T2 = \begin{pmatrix} T1 & 1 \\ T2 & 4 \\ T3 & 2 \end{pmatrix}, T3 = \begin{pmatrix} T1 & 3 \\ T2 & 1 \\ T3 & 5 \end{pmatrix}$$

Vector Clock Algorithmus mit Data Race aus Beispielcode.



Mit der WPF Applikation werden Programme instrumentiert und die Data Races dargestellt.

Ausgangslage: Die Arbeit «Dynamic Parallel Checker» behandelt die Entwicklung und Implementation eines Algorithmus zur Erkennung von Nebenläufigkeitsfehlern (Race Conditions) während der Laufzeit (dynamisch).

Vorgehen/Technologien: Der entwickelte Algorithmus basiert auf dem Vector Clock Algorithmus von Colin J. Fidge und Friedmann Mattern. Mit dem Vector Clock Algortihmus ist es möglich, nebenläufige Schreib- und Lesezugriffe in eine partielle Ordnung zu bringen, die sog. Happend-Before Beziehung. Locks und Unlocks sowie Thread oder Task Starts und Joins implizieren dabei eine Synchronisation der Vector Clocks zwischen den involvierten Threads bzw. Tasks, also eine Happend-Before Beziehung. Pro Thread oder Task werden alle nötigen Ereignisse, Zugriffe sowie Synchronisationen, in einer History protokolliert. Dies erlaubt es, anschliessend Data Races zu identifizieren: Ein Data Race liegt vor, wenn Zugriffe ohne Happened-Before Beziehung mit mindestens einer Schreiboperation auf dieselben Ressourcen stattgefunden haben.

Ergebnis: Die Implementation des Algorithmus instrumentiert Microsoft Intermediate Language Code (MSIL) mit Hilfe der Mono Cecil Bibliothek. Daher ist es möglich, mit dem Dynamic Parallel Checker alle für das .NET-Framework entwickelten Applikationen zu überwachen. Der bestehende MSIL Code wird um Codestellen erweitert, welche die Dynamic Parallel Checker Library aufrufen und mit Informationen beliefern. Diese Informationen werden schliesslich von Algorithmus verwertet, um Data Races zu detektieren.