Schnelle und präzise Data Race Erkennung zur Programmierzeit

Luc Bläser

Hochschule für Technik Rapperswil

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

HSR Concurrency Lab
Prof. Dr. Luc Bläser

KPS 2019
23-25.9.2019, Baiersbronn

Motivation

Nebenläufigkeitsfehler in der IDE erkennen:

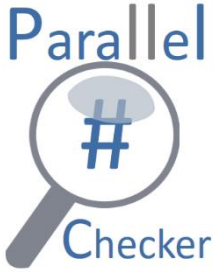
- Interaktiv während dem Codieren markieren
- Primärer Fokus auf Data Races

Anforderungen

- **Statisch**: Quellcode analysieren, evtl. unvollständig
- **Schnell**: Rückmeldung in ein paar Sekunden
- **Präzis**: So wenig Falschmeldungen wie möglich

Kompromiss: Findet nicht alle Fehler (unvollständig)

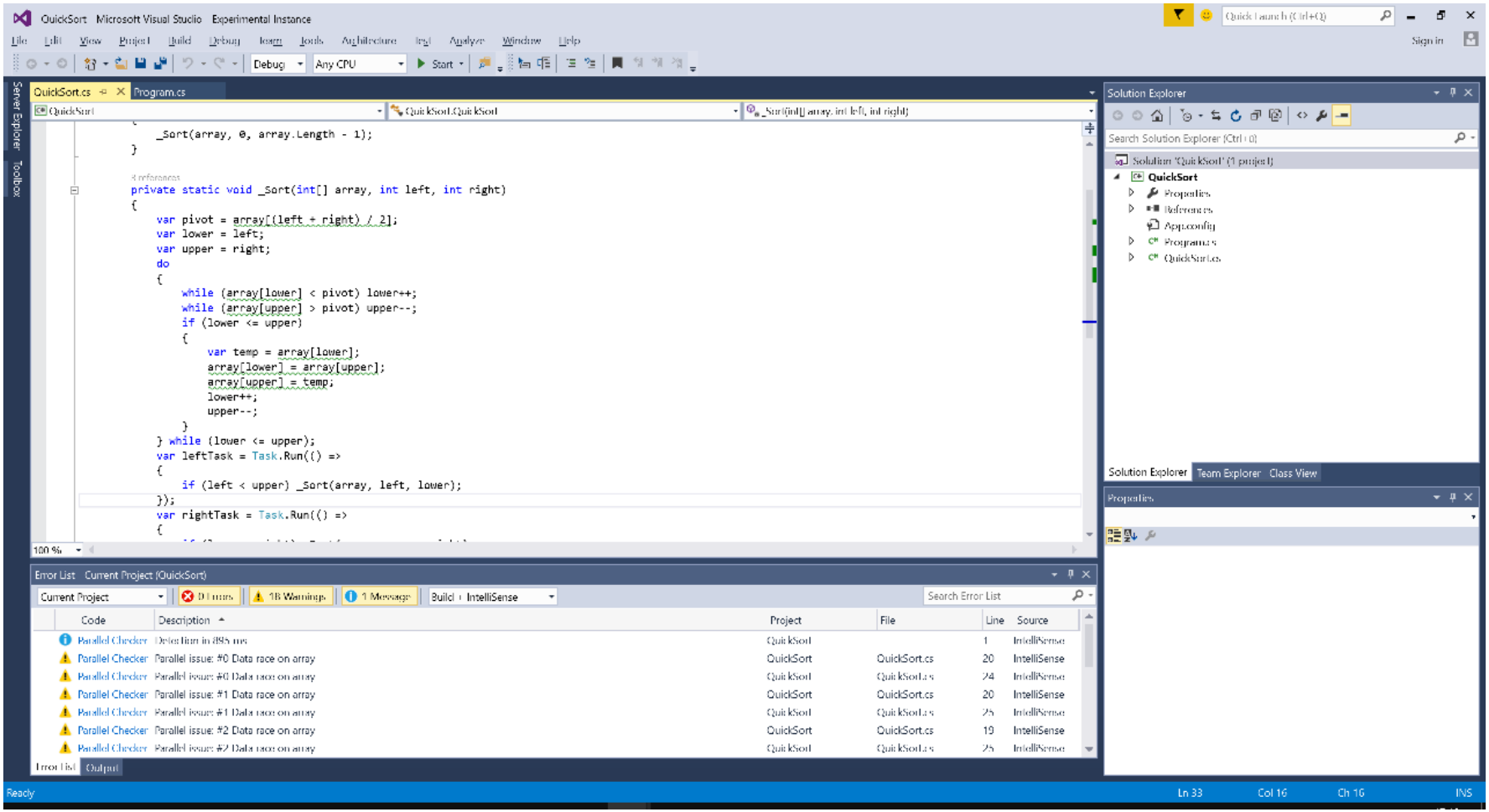
HSR Parallel Checker



- Neuer statischer Checker für Visual Studio
- Für neustes C#, breite Nebenläufigkeitsspektrum
  - Tasks, Async/Await, parallele Loops, diverse Synchronisations-Konstrukte, atomare Operationen, volatile, Finalizers, Timers, Parallel Queries …
  - UI-Anwendungen/Bibliotheken/Unit Tests/Console-App
- Verfügbar auf Visual Studio Marketplace (>1.6k installs, >4.5k downloads)

[ISSTA18] L. Bläser. Practical Detection of Concurrency Issues at Coding Time. International Symposium on Software Testing and Analysis (ISSTA) 2018.

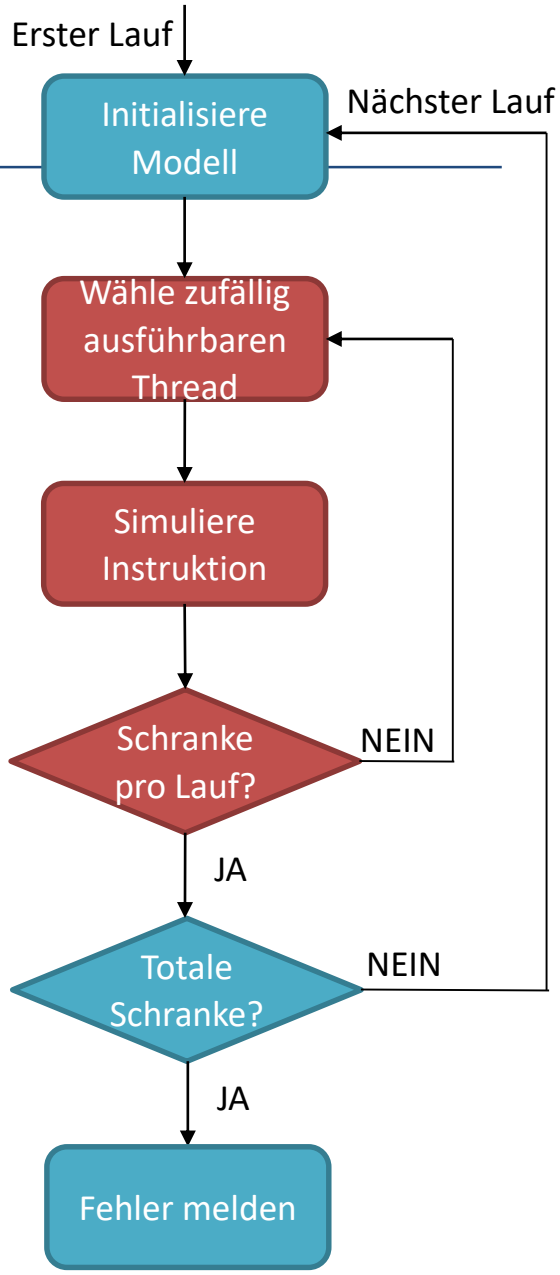
Kurze Vorführung



Ansatz

Randomisierte weitgehende konkrete Interpretation

- Auf internes Laufzeitmodell abbilden
- Ausführung auf Model simulieren
- Möglichst exakter Zustand mitführen
- Wiederholtes zufälliges Scheduling
- Schranken pro Lauf und Total
- Gefundene Fehler melden
- Vektoruhr (Epochs) für Data Races



Besondere Aspekte

- Reproduzierbarkeit der Resultate
  - Pseudo-Zufallszahlen mit Seed
  - Schranken in logischer Anzahl der Schritte und Größe
- Dynamische Technik in statischem Umfeld
  - Führt den Code nicht aus
  - Code kann unvollständig und inkorrekt sein
- Bewusst einfaches Modell
  - Zufälliges Scheduling, kein Constraint Solver
  - Analysiere mehr Code mit günstigem Verfahren

Abstrakte Zustände

- Mit unbekannten Input umgehen
  - Kommandozeilen-Argumente, Benutzer/Dateieingaben etc.
- Unbekannter Wert
  - Steht für beliebigen möglichen Wert
  - Propagiert durch die Ausdrücke
- Unpräzise Annahmen
  - Wähle zufällige Verzweigung bei unbekannter Bedingung
  - Ignoriere Locks, Thread Start/Join auf unbekanntem Objekt
  - Melde keine Data Races auf unbekannten Adressen

Ursache für False Positives (und False Negatives)

Experimentelle Auswertung

- 10 C# GitHub Projekte nach User Ranking
- 3 C# GitHub Projekte, «Concurrency» Tag
- 402 Assemblies
- 3.4 MLOC Quellcode

Project	Lines of Code	Assemblies
Roslyn 15.2	1,851,645	114
SignalR 2.2.2	86,574	31
Nancy 2.0.0	72,345	56
ILSpy 2.4	279,432	14
CefSharp 57.0.0	14,116	9
ReactiveUI 7.4.0	33,381	10
MsBuild 15.1.1012	397,281	20
Hangfire 1.6.14	73,986	12
Polly 5.2.0	91,363	6
NLog 4.4.11	63,381	6
Orleans 1.4.2	137,695	29
Akka.NET 1.2.2	225,744	82
Rx.NET 3.1.1	155,358	13
	3,482,302	**402**

Experimentelle Resultate

Analysierte Assemblies	402
Analyse-Zeit	13 Min. total
Zeit pro Assembly	1.7 Sek. Im Durchschnitt
Erkannte Probleme	121 Races
False Positives	14 (12%)
Echte Fehler	107
Produktive Fehler	89
Gefunden in	Roslyn, SignalR, NLog, Rx.NET

Gefundene Fehler

Roslyn

```
private static int s_delayMilliseconds = 0;

static GCManager() {
   System.Threading.Tasks.Task.Run(() => {
      …
      s_delayMilliseconds = (int)key.GetValue(…);
      …
    });
    …
}

internal static void TurnOffLowLatencyMode() {
  // same for UseLowLatencyModeForProcessingUserInput()
  if (s_delayMilliseconds <= 0) …
  …
}
```

data race

Und weitere Fehler…

SignalR

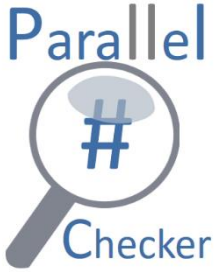
```
class Client {
  public static void Main() { …
    if (Arguments.IsController) {
      ControllerHub.Start(Arguments);
    }
    Run().Wait();
  }

  static async Task Run() {
    …
    while (TestPhase != ControllerEvents.Connect) {
      …
    }
    …
  }

  static void OnPhaseChanged(ControllerEvents phase) {
    …
    TestPhase = phase;
    …
  }
}

class ControllerHub {
  internal static void Start(…) {
    … ThreadPool.QueueUserWorkItem(_ => Run());
  }

  static void Run() {
    … RunConnect(); …
  }

  static void RunConnect(){
    … SignalPhaseChange(ControllerEvents.Connect); …
  }

  static void SignalPhaseChange(ControllerEvents phase) {
    … Client.OnPhaseChanged(phase);
  }
}
```

data race

Schlussfolgerung

- Nebenläufigkeit zur Programmierzeit prüfen
  - Direkt in IDE warnen, wenn Races programmiert werden
  - Erfordert statische, schnelle und präzise Erkennung
- Vollwertige C# Implementierung
  - Breites Spektrum an Nebenläufigkeits-Konstrukten
  - Einziger statischer Race Checker für modernes C#
- Einfacher, aber experimentell wirksamer Ansatz
  - Ebenso auf andere Sprachen anwendbar

Weitere Informationen



- **Paper (Open Access)**
  - L. Bläser. Practical Detection of Concurrency Issues at Coding Time. International Symposium on Software Testing and Analysis (ISSTA) 2018. ACM Digital Library.
- **Projekt-Webseite**
  - http://parallel-checker.com
- **Visual Studio Marketplace**
  - „HSR Parallel Checker for C# 8 (VS 2019)“