Compilerbau für die Common Language Runtime

Übersicht

© 2006 Martin v. Löwis

Common Language Runtime

- Terminologie: CLR, CLI, CLS, CTS, .NET
- Besser: “Compilerbau für die Common Language Infrastructure”?
- Compilerbau **mit** .NET: Formulierung von Compilern in “.NET-Sprachen” (C#)
  - ANTLR, (Phoenix?)
- Compiler **“nach”** .NET: Formulierung von Compilern, die .NET-Assemblies (MSIL) produzieren
  - C#-Compiler, ...
  - MSIL-Backend für GCC (Google Summer of Code)
- Compiler **“von”** .NET: Entwicklung von Compilern, die .NET-Assemblies konsumieren
  - JIT, Lego.NET
  - Fokus der Veranstaltung

© 2006 Martin v. Löwis

GCC

- ursprünglich: GNU C Compiler
  - veröffentlicht 1987
  - flexible Architektur zur Integration verschiedener Prozessorarchitekturen (portable compiler): backends
- GCC 2 (1992): Renovierung mit dem Ziel, verschiedene Eingabesprachen (frontends) zu erlauben
  - zunächst nur C++
  - NeXT/Apple entwickelt Objective C auf Basis von GCC, andere frontends für Ada, Fortran
  - kommerziell erweitert u.a. durch Cygnus Solutions
  - Streit um Entwicklungsmodell: EGCS
- GCC 3 (2001): GNU Compiler Collection
  - Ergebnis des EGCS-Projekts
- GCC 4 (2005): Erweiterung/Umbau der Codetransformationen (middle end, sic)

© 2006 Martin v. Löwis

Unterstützte Architekturen

- Alpha, ARC, ARM, AVR, Blackfin, CRIS, CRX, FRV, H8/300, HPPA, i386, AMD64, IA64, M32C, M32R/D, 68000, MCore, MIPS, MMIX, MN10300, MT, PDP-11, rs6000/PowerPC, S/390+zSeries, SH, SPARC, V850, VAX, Xtensa

© 2006 Martin v. Löwis

Unterstützte Frontends

- Ada, C, C++, Fortran 95, Java, Objective C, Objective C++
- verfügbare Frontends: Pascal, Modula-2, Modula-3, Mercury, VHDL, PL/I, .NET

© 2006 Martin v. Löwis

Lego.NET

- entwickelt von Jan Möller, Martin v. Löwis
- ursprüngliche Zielplattform: Lego Mindstorm (RCX)
  - Prozessor H8/300
  - 32kiB ROM, 32kiB RAM, 16 MHz, programmierbare Ports und Timer
  - IR-Schnittstelle
  - zuschaltbare Sensoren (Berührung, Licht, Drehung, Temperatur) und Aktoren (Motoren)
- Lego bietet Entwicklungsumgebung auf Basis von Lego Bytecode
  - NQC (Not Quite C) übersetzt NQC nach Lego Bytecode
- Alternative Betriebssystem: brickOS (C), leJOS (Java)
- Ziel: Ausführung von .NET-Code auf RCX unter brickOS

© 2006 Martin v. Löwis

Realtime .NET

- In Entwicklung: Andreas Rasche, Martin v. Löwis
- Ziel: Bereitstellung von .NET-Hochsprachen (C#, ...) für Echtzeitprogrammierung
- Spezifikation (API) und Implementierung
- Anforderungen an Implementierung:
  - Einhaltung von Zeitgrenzen (WCET: Worst-Case Execution Time)
    - auf Basis von IL-Anweisungen; WCET von Methoden kann daraus manchmal theoretisch abgeleitet werden
    - Nicht-Einmischung des Garbage-Collectors
    - Nicht-Einmischung des JIT
    - Zugriff auf Scheduler (Prioritäten) des Betriebssystems
  - Effizienz der Implementierung (oft auch in Bezug auf Stromverbrauch)
  - Portierbarkeit auf “verrückte” Zielplattformen

© 2006 Martin v. Löwis

Projektthemen

- Erweiterung des Compilers um das Interface-Konzept
- Erweiterung des Compilers um Ausnahmebehandlung
- Erweiterung des Compilers um Reflection-Information (und API)
- Integration in das Mono-Laufzeitsystem

© 2006 Martin v. Löwis

Interfaces

- Ziel: Das folgende Programm soll funktionieren:

```
interface A{
    int f();
}
class C:A{
    int f(){return 42;}
}
... public static void Main(){
    A a = new C();
    System.Console.WriteLine(a.f());
    }
```

© 2006 Martin v. Löwis

Interfaces (2)

- Herausforderung: Implementierung von Dispatch-Code für Interfaces
- Implementierungsstrategie in Anlehnung an Mono:
  - Auffinden der Zielmethode in konstanter Zeit
- Optionale Aspekte: casts und Typtests (isinstance)

© 2006 Martin v. Löwis

Ausnahmebehandlung

- Ziel: Das folgende Programm soll funktionieren

```
class A{
    static void f(){
        throw new ApplicationException(“Hello, world”);
    }
    public static void Main(){
        try{
            f();
        }catch(Application e){
            System.Console.WriteLine(e.Message);
        }
    }
}
```

© 2006 Martin v. Löwis

Ausnahmebehandlung

- Herausforderung: Integration von Ausnahmeregionen in GCC-Datenstrukturen
  - GCC sollte dann nötiges Unwinding generieren
- Optionale Aspekte:
  - Hierarchische Ausnahmebehandlung
  - VM-Ausnahmen (NullReferenceException, IndexOutOfRangeException)
  - Untersuchung der Zeitkomplexität
  - Mono-Kompatibilität

© 2006 Martin v. Löwis

Reflection

- Ziel: Das folgende Programm soll funktionieren

```
class A{
    public static void Main(){
        System.Console.WriteLine(typeof(System.Console).FullName);
    }
}
```

- Herausforderung: Generierung von Mono-konformen Tabellen, die die benötigten Informationen enthalten
- Optional: Aufruf von Methoden über Reflection

© 2006 Martin v. Löwis

Mono-Integration

- Ziel: Das folgende Programm soll sich unter Linux|Windows|OSX mit einer abgerüsteten Version von Monos mscorlib.dll zu einem komplett zielabhängigen Programmimage linken lassen

```
class A{
    public static void Main(){
        System.Console.WriteLine("Hello, World");
    }
}
```

- Lego.NET ist bisher teilweise "mono-kompatibel"
  - Layout von System.Object, System.Array, System.String folgt Mono-Layout
  - Kompatibilität der "calling convention" unbekannt
  - bisher keine Integration von Monos VES (virtual execution system)

© 2006 Martin v. Löwis

Mono-Integration

- Herausforderung: Abrüstung von Monos mscorlib so weit, dass GCC sie übersetzen und linken kann
- Lösungsstrategie: Schrittweiser Verzicht auf alles, was für das Testprogramm nicht zwingend erforderlich ist; Realisierung der verbleibenden fehlenden Teile im GCC

© 2006 Martin v. Löwis

Themen der Vorlesungen

- Aufbau des GCC, Arbeit mit den GCC-Quellen
- Aufbau der Common Language Infrastructure und des MSIL-Bytecodes
- Implementierung objektorienter Konzepte (in Mono)
- Funktionsweise des IL-Frontends
- weiterhin:
  - Source Code Analyse (Parsing, semantische Analyse)
  - Synthese von Maschinencode (GCC Machine Descriptions)
  - Optimierungspässe (GIMPLE, SSA, peep-hole, ...); Optimierungspotentiale in OO-Sprachen

© 2006 Martin v. Löwis

Erste Schritte im Projekt

- Zugriff auf GCC-Quellen (https://www.dcl.hpi.uni-potsdam.de/projects/gcc/trunk; anerkanntes Zertifikat erforderlich)
- Übersetzung des IL-Compilers für eine Zielplattform (etwa: Linux/x86)

© 2006 Martin v. Löwis