Projektarbeit Pony

Atay Oezcan und Daniel Weingand

12. Januar 2026

Inhaltsverzeichnis

1 Einleitung

Diese Projektarbeit befasst sich mit der Sprache Pony und und untersucht, wie es mit einigen bekannten Problemen, wie z.B. Nebenläufigkeit, umgeht. Zunächst soll die Sprache und seine Konzepte erklärt werden und anschließend erfolgt ein Vergleich mit den Swift Aktoren. Im Kapitel Projektstruktur findet sich die allgemeine Struktur des Ordners.

2 Pony

Pony ist eine objektorientierte open-source Programmiersprache mit Aktormodell, das 2015 veröffentlicht wurde. Die Sprache verspricht gute Performance für Anwendungen mit hoher Parallelität ohne deadlocks, oder anderen typischen Problemen, unter denen die meisten Programmiersprachen leiden.

2.1 Pony Features

Pony bietet einige Features, die es dem Programm ermöglichen performant und sicher zu laufen.

2.1.1 Aktormodell

Aktoren sind Modelle für nebenläufige Programme, die ausschließlich asynchron Nachrichten austauschen. Das sorgt dafür, dass der Aktor nicht auf eine Nachricht warten muss und stattdessen weiter agieren kann. Die Nachrichten müssen nicht in der Reihenfolge ankommen, in der sie versendet werden. Jeder Aktor verfügt dabei über seinen eigenen Speicherbereich, den er mit keinem anderen Aktor teilt. Da Aktoren Nachrichten asynchron abarbeiten und keine anderer Prozess Zugriff auf den Speicherbereich hat, ist der Ablauf innerhalb des Aktors sequenziell. Der Nachteil ist, dass Informationen mehrfach vorliegen können, da kein Speicherbereich geteilt werden kann.

2.1.2 Typensicher

Die Typen in Pony sind statisch, d.h. der Typ der Variable ist zur Kompilierung bekannt und daher werden Fehler zur Compilezeit erkannt. Gegenbeispiele hierzu sind Python und JavaScript, die erst zur Laufzeit den Typ prüfen.

2.1.3 Speichersicher

Pony hat keine Buffer, Pointer oder NULL.

2.1.4 Keine Exceptions

Es gibt keine Laufzeitfehler. Über die Semantik werden Fehler beim Kompilieren erkannt. Sobald ein Programm erfolgreich kompiliert können keine Laufzeitfehler mehr entstehen.

2.1.5 Keine data-races

Jeder Aktor steht für einen Thread und das Typsystem von Pony stellt zur Kompilierungszeit sicher, dass keine data-races auftreten.

2.1.6 Keine Deadlocks

Es gibt keine Locks, da die Aktoren keinen geteilten Speicherbereich haben und innerhalb der Aktoren Anweisungen sequenziell laufen. Pony erlaubt nur die Weitergabe von unveränderlichen Daten über Referenzen

2.1.7 Nativer Code

Pony hat einen Compiler, der den code zunächst in Zwischencode umwandelt und optimiert.

2.1.8 C-kompatibel

Pony Programme können in C-Code übersetzt werden.

2.2 Nachteile

Da Pony sehr neu und wenig verbreitet, kommt die Nutzung mit einigen Nachteilen.

2.2.1 Mangelnde API-Stabilität

Pony hat die Version 1.0 noch nicht erreicht.

2.2.2 Mangel an hochwertigen Bibliotheken von Drittanbietern

Folgende Bibliotheken existieren aktuell für Pony:

- appdirs
- fork_join
- github_rest_api
- glob
- http

- http_server
- json
- logger
- lori
- peg
- postgres
- reactive_streams
- regex
- semver
- ssl
- templates
- valbytes

2.2.3 Begrenzte native Werkzeuge

Keine IDE unterstützt die Pony Language. Auch Analyse Tools sind derzeit nicht verfügbar.

2.3 Pony Aktoren vs. Swift Aktoren

Swift ist eine Allzweck-Programmiersprache, die 2014 von Apple veröffentlicht wurde. und mehrere Konzepte aus anderen bekannten Sprachen wie C++, Java etc. besitzt. Es kann zur Entwicklung von Apps, Systemsoftware, Cloud-Diensten und eingebetteter Firmware verwendet werden und läuft unter Windows und MacOS.

Merkmal	**Pony Actors**	**Swift Actors**
Kernparadigma	Actor Model	Concurrency Model mit Actor-Design
Daten	Unveränderlich (immutable)	Veränderlich, aber nur ein Thread darf Daten ändern
Kommunikation	Nachrichtensystem zwischen Akteuren	Asynchrone Methodenaufrufe für Datenmanipulation
Fehlertoleranz	Hohe Fehlertoleranz durch Isolation der Akteure	Keine spezifische Fehlertoleranz, aber sicherer Zugriff
Zugriffssteuerung	Nachrichtenbasierte Kommunikation	Asynchrone Synchronisation für gleichzeitigen Zugriff
Skalierbarkeit	Sehr hoch, mit effizienter Nachrichtenzustellung	Hoch, durch optimierten Zugriff auf Threads
Speicherverwaltung	Automatische Speicherverwaltung	Automatische Speicherverwaltung
Ziel	Stark auf Parallelität und Fehlertoleranz fokussiert	Fokus auf sichere und einfache Konkurrenz ohne Datenkorruption

Tabelle 1: Vergleich zwischen Pony Actors und Swift Actors

3 Projektstruktur

Das Projekt besteht aus den Themen der folgenden Unterkapitel und enthalten jeweils ein README zum Thema und weitere README zu jeder Programmiersprache. Es werden C++, Rust, Swift, GO und Pony untersucht.

3.1 data-races

Ein Data Race ist ein Fehler, der in nebenläufigen oder parallelen Programmen auftrete kann. Er tritt auf, wenn:

1. Mindestens zwei Threads/Tasks gleichzeitig auf dieselbe Speicherstelle zugreifen
2. Mindestens einer davon schreibt
3. Keine Synchronisation vorhanden ist (z. B. Lock, Actor, Mutex)

3.2 deadlocks

Ein Deadlock ist ein Zustand in nebenläufigen Programmen, bei dem sich mehrere Threads gegenseitig blockieren und keiner weitermachen mehr weitermachen kann.

1. Mehrere Threads Ressourcen halten
2. Threads auf Ressourcen warten, die von den anderen gehalten werden

3.3 dining-philosophers

Die Dining Philosophers sind ein klassisches Gedankenexperiment aus der Informatik, die das Probleme der Nebenläufigkeit veranschaulicht.

Grundidee
Mehrere Philosophen sitzen an einem runden Tisch, auf dem Essen liegt. Zwischen je zwei Philosophen liegt eine Gabel und jeder Philosoph braucht zwei Gabeln, um essen zu können. Die Philosophen wechseln zwischen denken und essen.

Das Problem
Wenn alle Philosophen gleichzeitig ihre linke Gabel aufnehmen und dann auf die rechte Gabel warten, dann erhält jeder eine Gabel und wartet auf die andere. Niemand kann essen.

3.4 memory-leaks

Ein Memory Leak ist ein Problem in Programmen, bei dem Speicher im Hauptspeicher reserviert, aber nie wieder freigegeben wird, obwohl er nicht mehr benötigt wird. Das führt dazu, dass das Programm mit der Zeit immer mehr Speicher verbraucht, bis das System langsam wird oder abstürzt.

3.5 papers

Hier sind zwei Paper der Erfinder von Pony, die einen tieferen Einblick in seine Konzepte und Hintergründe gewähren.

3.6 runtime-errors

Ein Runtime Error ist ein Fehler, der während der Ausführung eines Programms auftritt, also nicht schon beim Kompilieren erkannt wird. Im Gegensatz zu Syntaxfehlern (Compile-Time Errors) treten Runtime Errors erst auf, wenn der fehlerhafte Code tatsächlich ausgeführt wird.

3.7 server-http

Dieser Ordner enthält eine einfache Implementierung eines Servers, der HTTP Anfragen annehmen kann.

3.8 server-tcp

Dieser Ordner enthält eine einfache Implementierung eines Servers, der über TCP kommuniziert.

3.9 Vorstellung

In diesem Abschnitt findet sich ein kurzes Beispielprogramm, das Klassen, interface und Vererbung in Pony zeigt.