Space Optimizations in Deterministic and Concurrent Call-by-Need Functional Programming Languages

Abstract

Nils Dallmeyer

In dieser Dissertation werden Speicherverbrauch und Laufzeit von lazy-auswertenden funktionalen Programmiersprachen untersucht.

Die getypten erweiterten λ-Kalküle LRP und CHF* werden als Kernsprachen für Haskell und Concurrent Haskell verwendet. Jeweils zu LRP und CHF* kompatible abstrakten Maschinen werden vorgestellt.

Zur Störungsfreiheit der Platzmessung wird ein klassisch implementierbarer Garbage Collector nach jedem LRP-Rechenschritt angewendet. Die Größe `size` von Ausdrücken, sowie das Platzmaß *spmax* als Maximum aller garbage-freien Ausdrücke während einer LRP-Programmausführung, werden definiert.

Programm-Transformationen werden in Form von Code-zu-Code-Transformationen betrachtet. Die Begriffe *Space Improvement* und *Space Equivalence* als Eigenschaften von Transformationen werden definiert. Ein Space Improvement verändert weder die Semantik noch erhöht sich der benötigte Speicherverbrauch, bei einer Space Equivalence bleibt der Speicherverbrauch gleich. Einige Transformationen werden als Space Improvements und Equivalences gezeigt.

Die abstrakte Maschine $\mathrm{M1}_{sp}$ für Messungen des Speicherverbrauchs eingeführt. Eine Implementierung der $\mathrm{M1}_{sp}$ wird für komplexere Platz- und Laufzeit-Analysen genutzt.

Total Garbage Collection ersetzt Teilausdrücke durch eine nicht-terminierende Konstante mit Größe Null, falls sich an der Gesamtterminierung nichts ändert. Dadurch wird die Unabhängigkeit des Improvement-Begriffs vom verwendeten Garbage Collector gesteigert.

Analog zu Space Improvements und Equivalences werden die Begriffe *Total Space Improvement* und *Total Space Equivalence* definiert, bloß dass diese den Total Garbage Collector bei der Platzmessung nutzen. Einige Total Space Improvements und Equivalences werden gezeigt.

Platzmaße für CHF* werden definiert, die kompatibel zu den Platzmaßen von LRP sind. Ein Algorithmus wird entwickelt, der den benötigten Speicherplatz für unabhängige Prozesse mit synchronen Start- und Endpunkten bezüglich Scheduling in Sortierkomplexität berechnet. Bei Hinzunahme einer konstanten Anzahl an Synchronisierungen und konstanter Anzahl an Prozessen ist die Laufzeit polynomiell, bei beliebigen Synchronisierungsbeschränkungen ist das Problem NP-vollständig.

Für Speicher- und Laufzeitanalysen in CHF* werden abstrakte Maschinen entworfen und Implementierungen dieser für Speicher- und Laufzeit-Analysen verwendet.

Space Optimizations in Deterministic and Concurrent Call-by-Need Functional Programming Languages

Abstract

Nils Dallmeyer

In this thesis the space consumption and runtime of lazy-evaluating functional programming languages are analyzed.

The typed and extended λ-calculi LRP and CHF* as core languages for Haskell and Concurrent Haskell are used. For each LRP and CHF* compatible abstract machines are introduced.

Too lower the distortion of space measurement a classical implementable garbage collector is applied after each LRP reduction step. Die `size` of expressions and the space measure *spmax* as maximal size of all garbage-free expressions during an LRP-evaluation, are defined.

Program-Transformations are considered as code-to-code transformations. The notions *Space Improvement* and *Space Equivalence* as properties of transformations are defined. A Space Improvement does neither change the semantics nor it increases the needed space consumption, for a space equivalence the space consumption is required to remain the same. Several transformations are shown as Space Improvements and Equivalences.

The abstract machine $M1_{sp}$ for space measurements is introduced. An implementation of $M1_{sp}$ is used for more complex space- and runtime-analyses.

Total Garbage Collection replaces subexpressions by a non-terminating constant with size zero, if the overall termination is not affected. Thereby the notion of improvement is more independent from the used garbage collector.

Analogous to Space Improvements and Equivalences the notions *Total Space Improvement* and *Total Space Equivalence* are defined, which use Total Garbage Collection during the space measurement. Several Total Space Improvements and Equivalences are shown.

Space measures for CHF* are defined, that are compatible to the space measure of LRP. An algorithm with sort-complexity is developed, that calculates the required space of independent processes that all start and end together. If a constant amount of synchronization restrictions is added and a constant number of processors is used, the runtime is polynomial, if arbitrary synchronizations are used, then the problem is NP-complete.

Abstract machines for space- and time-analyses in CHF* are developed and implementations of these are used for space and runtime analyses.