Strikte Zyklische Berechnungen: Von der Theorie zur Praxis

(Erweiterte Zusammenfassung)

Baltasar Trancón y Widemann

Die traditionelle initiale (algebraische) Semantik rekursiver Datentypen verbietet unendliche und zyklische Daten. Dadurch wird gewährleistet, daß rekursive Traversierungen terminieren. In der dualen finalen (coalgebraischen) Semantik dagegen sind diese gestattet. Während Unendlichkeit nur approximativ durch nichtstrikte Semantik und *lazy*-Auswertung modelliert werden kann, lassen sich Zyklen in der Praxis leicht durch wechselseitige Zuweisung erzeugen. In imperativen Sprachen mit expliziten Zeigern sind einfache zyklische Datenstrukturen, wie etwa Ringlisten, durchaus bekannt. Die Termination einer Traversierung kann durch Erkennung des Zyklus gesichert werden. Sind aber im Allgemeinen die Eintrittspunkte von Zyklen nicht bekannt, müssen besuchte Zellen markiert werden.

Wir untersuchen ein allgemeines Verfahren zur Traversierung zyklischer Daten, bei dem die Markierung und Erkennung von Zyklen dynamisch durch Suche auf dem Aufrufstack geleistet wird. Damit lassen sich einerseits bei geeigneten Aufrufkonventionen corekursive Funktionen implementieren, die einen Zyklus in der Eingabe auf einen Zyklus in der Ausgabe abbilden. Diese können in strikter Semantik, also *eager* ausgewertet werden. Andererseits kann dieselbe Technik der Zyklenerkennung auch zur Implementierung kleinster und größter Fixpunkte von rekursiven Prädikaten auf zyklischen Daten genutzt werden. Anders als in einem *lazy*-Kontext lassen sich damit auch Probleme entscheiden, die nicht durch einen endlichen Präfix der Eingabe determiniert sind. Die Kombination von zyklischen Funktionen und Prädikaten stellt somit ein sehr mächtiges Werkzeug für die funktionale Verarbeitung zyklischer Daten dar.

Die Behandlung von Zyklen in corekursiven Funktionen erfordert eine besondere Vorgehensweise. Anders als in der algebraischen Sichtweise, bei der Berechnung durch Reduktion von unten nach oben realisiert wird, ist es hier erforderlich, von oben nach unten vorzugehen. Insbesondere muß in jeder Inkarnation die Wurzelzelle des Ergebnisses festgelegt werden, bevor ein corekursiver Abstieg erfolgt, damit im Falle eines erkannten Eingabezyklus auf diese verwiesen werden und so ein Ausgabezyklus erzeugt werden kann. Die Initialisierung des Ergebnisses erfolgt dann zweckmäßigerweise nicht durch Einsammeln der corekursiven Teilergebnisse, sonder durch Delegation an die corekursiven Aufrufe. Eine operationale Semantik dieses Vorgehens läßt sich also eher mit dem Mittel der Zuweisung als mit dem der kontextfreien Reduktion formulieren.

Wir stellen eine abstrakte Maschine vor, die diesen Anforderungen genügt, und die über Mittel zur Erkennung und Behandlung von Zyklen verfügt. Die Erzeugung von Daten erfolgt durch Allokation von Zellen und explizite Zuweisung von Referenzen, die Freigabe bleibt implizit. Somit ähneln die Maschinenprogramme denen objektorientierter Maschinen wie der Java VM. Genuin funktionale Mechanismen wie parametrische Polymorphie und Funktionen höherer Ordnung erweitern die Mächtigkeit der Maschine zu einer vollwertigen Zielplattform für die Übersetzung einer noch zu entwerfenden zyklischen funktionalen Programmiersprache. Die Implementierung von Werkzeugen rund um die Maschine umfaßt Assembler-Sprache, Assembler, Interpreter, optimierenden Compiler und *just-in-time*-Compiler. Die Implementierung ist in Java geschrieben und besitzt eine grafische Oberfläche, die das Experimentieren mit allen Werkzeugen ermöglicht.

Neben der technischen Umsetzung zyklischer Berechnungen untersuchen wir auch die Theorie zyklischer Datentypen. Dazu wird das konkrete Beispiel zyklischer Listen untersucht. Für diese stellen wir eine kanonische endliche Normalform, nämlich die Zerlegung in *Präfix* und *Periode*, und effektive Regeln zu deren Berechnung vor, und klassifizieren die traditionellen Datentypen der endlichen und der Ringlisten als Spezialfall mit leerer Periode beziehungsweise leerem Präfix.