tSig: typisierte Signalerzeugung
Über den Datenfluss und in die Syntaxwälder

Baltasar Trancón y Widemann[ab]
Markus Lepper[b]

[a]Universität Bayreuth
[b]<semantics/> GmbH

Zusammenfassung. In der Praxis der digitalen Erzeugung und Verarbeitung von Audiosignalen sind eine Reihe von Werkzeugen teils seit Jahrzehnten etabliert, wie *C-Sound*, *Max*, *Supercollider*, etc. Diese Systeme und ihre diversen Bibliotheken beinhalten einerseits beachtliches Domänen-Wissen im Bereich der Audiosignalverarbeitung, sind aber andererseits von zeitgemäßem professionellem Sprachdesign völlig unberührt.

Dies hat konkrete schädliche Auswirkungen bezgl. Wiederverwendbarkeit, Wartbarkeit, Dokumentierbarkeit, Fehlersuche, etc. Besonders fehlende Typstrenge, die in allen Systemen verlangte Kombinierung disparater Programmierparadigmen (Stromverarbeitung, Ereignisse, imperativ realisierte Zustandsübergänge, etc.), das aus beidem folgende Fehlen jeder semantischen Fundierung, veraltete Konzepte von Namensräumen und Parametrisierung, etc. machen das Programmieren in diesen Systemen zu einer nervlichen wie ästhetischen Qual.

In neuerer Zeit gibt in der Tat einige Projekte, die diese Mängel beheben wollen und ausgehend von einer funktionsbasierten Deklaration von Signalflüssen den Code zur Signalverarbeitung direkt aus semantischen Modellen generieren (*Faust*, *grapefruit*, *feldspar*).

In dieser Tradition entwickeln wir *tSig* als Prototyp einer funktionalen Programmiersprache zur zeitgemäß abstrahierbaren und präzisen Beschreibung von Signalprozessen, mit dem Hauptziel von Realzeit-Audio-Verarbeitung, nach rigorosen Entwurfsprinzipien: Ein starkes statisches Typsystem soll Korrektheit und Modularisierung unterstützen, wobei benutzerdefinierte Datentypen, Ausnahmebehandlung und Parametrisierung auf die wohlbekannten Mittel Produkt, Coprodukt und Funktionstypen höherer Ordnung abgebildet werden. Die kompositionale Syntax erlaubt den Wechsel zwischen und die Kombination von funktionalem Stil, basierend auf Termen und pattern matching, mit der algebraischen Beschreibung von Datenflussgraphen. Die synchrone Semantik basiert an den Schnittstellen funktionaler Blöcke auf primitiv corekursiven stromverarbeitenden Funktionen, innerhalb eines Blocks auf einem relationalen Datenbankkalkül. Im Hinblick auf primitive Zielplattformen werden die Konstrukte höherer Ordnung in einer intensiven Transformations- und Optimierungsphase eliminiert. Als Zielplattform sollen je nach Anwendung Simulatoren, existierende Bibliotheken oder DSPs in Frage kommen.