Abstract (English)

Gradual typing is a promising technique which offers programmers the freedom to have both typed and untyped code working together seamlessly. However, little such gradual programming languages exists, because designing gradual typesystems is difficult. This is caused by both a lack of understanding of these typesystems and a lack of tools helping developing them.

If more gradual programming languages would exist, programmers would be more productive, as they are able to choose for each part of their code if it would benefit the most from a dynamic or static approach. On a longer timeframe, the programmer can migrate their codebase from or to a fully dynamic to a static one as is needed.

In Ronald Garcia's paper, *Abstracting Gradual Typing*, a way to gradualize typesystems is proposed. In this dissertation, a new tool is presented which implements the necessary abstract interpretation of this paper.

The introduced tool uses a new metalanguage to easily specify programming languages and helps to construct a gradual variant of the language. Apart from the gradualization, the tool offers support for general language design, such as interpretation of target programs, typechecking and property verification.

To verify the usefullness of the tool, a simple, functional language was implemented. During this development, many bugs were caught by the automatic checks. Based on the functional language, a gradual counterpart was developed with support of the tool.

Abstract (Nederlands)

Gradual typing is een veelbelovende techniek die het mogelijk maakt om in hetzelfde programma statisch getypeerde en dynamisch getypeerde stukken code te hebben, die zonder problemen met elkaar samenwerken. Jammergenoeg bestaan er maar weinig graduele programmeertalen, omdat het ontwerpen van graduele typesystemen een moeilijke taak is. Dit komt omdat graduele typesystemen slecht gekend zijn en omdat weinig tools beschikbaar zijn.

Als er meer graduele programmeertalen zouden bestaan, zou programmeren efficienter kunnen gebeuren: de programmeur zou voor elk stuk code kunnen kiezen tussen een statische of dynamische aanpak, al naargelang wat het voordeligste is voor elk stuk code - zo zou het stuk code dat constant wijzigt dynamisch getypeerd worden, waar het ontwerp van een moeilijk algoritme de hulp van een statische typechecker krijgt.

De paper *Abstracting Gradual Typing* van Ronald Garcia beschrijft stap voor stap hoe een statisch typesysteem kan worden omgebouwd tot een gradueel typesysteem. In deze masterthesis presenteren we een tool die de nodige abstracte interpretatie implementeert om deze techniek toe te passen. De tool gebruikt een nieuwe metataal die gebruikt kan worden om arbitraire programmeertalen te specifieren en biedt ondersteuning voor deze taak: uit de specificatie wordt automatisch een interpreter gebouwd, eigenschappen van de programmeertaal worden getest en verschillende controles helpen om kleine foute te detecteren.

Om de tool te evalueren werd een simpele functionele programmeertaal gebouwd. Tijdens het ontwerpen werden verschillende bugs gedetecteerd. Deze simpele functionele programmeertaal werd, met behulp van de tool, gegradualiseerd.