Deciding associativity in L0

Topics in Programming Languages
Datalogisk institut, Copenhagen University (DIKU)

Kristoffer Søholm & Oleksandr Shturmov
`{soeholm,oleks}@diku.dk`

May 12, 2013.

Abstract

In the context of parallelizable second-order array combinators, it is often necessary for the function argument to be an associative function. We describe a partial decider for the associative property of functions in the language L0, for implementation in the parallelizing compiler l0c, develped under the HIPERFIT project. The problem is undecidable in general. This paper explores the practical feasibility of such a decider in the context of function arguments to second-order array combinators.

Keywords: Assosiativity, parallelism, second-order array combinators, L0, l0c, HIPERFIT.

1 Introduction

Some parallelizing compilers will assume that the user has supplied an associative function. Others will require for the function to be explicitly marked as "associative". Others still will try to deduce the associativity property, and warn the user if the function clearly isn't associative.

1.1 An associative type

For ease of analysis, we introduce the concept of an *associative type* — a function type that unifies with the type of any associative function. An associative function consumes at least two values of type t, and returns a value of type t.

We refer to functions that unify with the associative type, but are not themselves associative as *not associative*. We refer to all other functions as *associative*.

In particular, we refer to functions that do not unify with an associative type as associative. This is to allow us to say that e.g. unary functions are associative. Although this does not make sense mathematically, our reasoning is that functions that do not unify with an associative type cannot be chained into a function whos type unifies with an associative type.

2 Builtins

2.1 Chaining

We consider first whether simply chaining applications of the operator is associative. That is, for each binary operator $\oplus : t \times t \rightarrow t$, we consider whether $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ by the L0 specification. Unary operators are trivially associative under chaining. For binary operators with arguments of two different types, as well as ternary and n-ary operators, it depends. Later, we'll allow intermixing of operators.

2.1.1 Arithmetic operators

L0 has a range of binary arithmetic operators, namely `+`, `*`, `-`, `/`, `pow`, and $\sim$, where the last operator is a unary negation operator. The operators defined on arguments of type either `int` or `real` (L0 is strongly-typed), yielding a value of the same type.

As of now, there are no restrictions in the L0 standard as to how `real`s are to be represented, so we confine our attention to `int`s. For `int`s we assume a p-bit two's complement representation.

Our observations are summarized in Table 1.

Operator	Chaining is associative
`+`	yes
`*`	yes
`-`	no
`%`	no
`pow`	no
~	yes*

Table 1: Associativity of chaining arithmetic operators.

2.1.2 Relational operators

L0 has a couple binary relational operators, namely `=`, `<`, and `<=`, having the obvious semantics. The operators are defined on arguments both of type either `int` or `real`, yielding a value of type `bool`.

Relational operators are not chainable in L0. These operators are only interesting in combination with other operators.

2.1.3 Logical operators

L0 has a couple logical operators, namely `&&`, `||`, and `not`, where the last is a unary negation operator. The operators are defined on arguments of type `bool`, yielding a value of type `bool`.

Our observations are summarized in Table 2.

Operator	Chaining is associative
`&&`	yes
`||`	yes
`not`	yes*

Table 2: Associativity of chaining logical operators.

2.1.4 Bitwise operators

L0 has a range of bitwise operators, namely `^`, `&`, `|`, `>>`, and `<<`, having semantics as in C. The operators are defined on arguments of type `int`, yielding a value of type `int`.

Our observations are summarized in Table 4.

Operator	Chaining is associative
`^`	yes
`&`	yes
`|`	yes
`>>`	yes
`<<`	yes

Table 3: Associativity of chaining bitwise operators.

2.1.5 Other functions

L0 has a range of other builtins, e.g. various second-order array combinators. Of particular interest to us are the functions `concat`, `min`, and `max`. The other builtins do not meet the requirement of unifiability of their type with an associative type, or fall prey to the fact that functions are not first-class citizens in L0. These operators are trivially associative.

Our observations are summarized in Table 4.

Operator	Chaining is associative
`concat`	yes
`min`	yes
`max`	yes

Table 4: Associativity of chaining bitwise operators.

2.2 Combining

Light's associativity test [Kehayopulu & Argyris].

*All unary operators are associative by definition.

References

[Kehayopulu & Argyris] Niovi Kehayopulu and Philip Argyris *An algorithm for Light's associativity test using Mathematica*. Journal of computing and information.3 (1): 87–98. ISSN 11803886