yalaa and P1788

Stefan Kiel

20 Mai 2011

Contents

1 Introduction

Currently there is a ongoing effort to standardize interval arithmetic (IA) in order to improve the hardware and software support and get increase the compatibility between different implementations. These efforts led to a working group for the *IEEE Standard For Interval Arithmetic P1788*. Altough P1788 is still a draft, does not focus on the programming level and does not consider other verified models like affine arithmetic (AA), it can still provide valuable input for implementing yalaa.

In this document we will explore which motions or parts of the P1788 are applicable for an AA library at programming language level and how we can incoperate them into yalaa.

2 Overview of Motions

The following tables lists the (currently passed) motions of P1788, which we consider relevant for the design of yalaa.

Number	Subject	Goal for yalaa
8	Exception Handling	Support via ErrorPolicy
10	Elementary Functions	Support the function set
18	Domain tretit and bool_set	Support domain_tretit

3 Required Operations

The P1788 standard requires that a conforming implementation supplies various operations and elementary functions. These specified in the current draft [1] §4.5 and in motion 8 [**?**]. The status of these operations is listed in the following Table.

P1788 also specifies a set of recommended functions. Currently there are no plans to support them in yalaa.

4 Exception Handling

The currently discussed exception handling approach in P1788 is based on decorations and trits [3], [2]. The basic idea is, that an interval is combined with some sort of *decoration* forming a *decorated interval*. A decoration is a tri-bool (true, false, unknown) (**Richtig???** Hayes erzählt in [**?**] teilweise was anderes). It allows the user to deduce whether an exception has occurred in the computational graph from which the decorated interval originated.

Motion 18 [2] requires a decoration, which indicates whether the natural domain of a called function has been violated. The current draft for P1788 specifies the flags *valid*, *defined*, *continous* and *bounded*. In its current form there are 20 possible[1] combinations. However, a voting on the final form is still outstanding.

In [**?**] Hayes describes another[2] set of attributes: *domain*, *defined and continuous* and *defined and bounded*, where domain is a tetrit and the latter two are booleans and are further mapped into one value, resulting in 5 possible decorations. At the moment, it is our goal to support this simpler approach in yalaa. The decorations and their meaning are outlined in the following table:

[1] i.e.: meaningful

[2] but very similar

Dec.	Meaning
$\mathbb{D}_4$	safe
$\mathbb{D}_3$	everywhere defined
$\mathbb{D}_2$	somewhere defined, somewhere undefined
$\mathbb{D}_1$	everywhere undefined
$\mathbb{D}_0$	ill-formed

These decorations can be mapped to the error flags which are propagated during yalaa's computation process:

Flag	Meaning	Mapping
`VALID`	No error	$\mathbb{D}_4$
`P_D_VIOL`	Partial violation of domain	$\mathbb{D}_2$
`C_D_VIOL`	Complete violation of domain	$\mathbb{D}_1$
`UNBOUNDED`	No finite bounds or overflow	$\mathbb{D}_3$
`ERROR`	Unknown error	$\mathbb{D}_0$

However, there is a small semantic difference. The `UBOUNDED` flag can be raised even if the function is bounded on $\mathbb{R}$ and continous if an overflow occurs. This contradicts Prop. 3 of [**?**]. Further the general error flag is mapped to $\mathbb{D}_0$, which may not be correct in all cases.

Problem: Ordnung, Hayes definiert in Prop. 4 eine Ordnung auf den $\mathbb{D}_i$ welche hier nicht stimmt. Tritt UNBOUNDED auf, so ist der affine Teil unbrauchbar während bei `P_D_VIOL` immerhin noch ein brauchbares Ergebnis produziert wird. Jedoch ist in diesem Fall die Domäne verletzt, im Fall von `UBOUNDED` nicht. Wie geht man mit diesem Fall um? Rein von den theoretischen Eigenschaften gilt die Ordnung, sie mappt aber nicht intuitiv auf den affinen Teil der dekorierten Form.

The mapping is implemented in through the policy class `ErrorPolP17188`.

References

[1] John Pryce (Tech. Eds.). P1788: IEEE Standard for Interval Arithmetic Version 02.2.

[2] N. T. Hayes. Trits to tretris. Technical report, Sunfish Studio, LLC, 2010. P1788, Motion 18.

[3] Nathan T. Hayes and A. Neumaier. Exception handling for interval arithmetic. P1788, Motion 8.

Table 1: Status of P1788 required operations in yala

Operation	Supported	Notes
sqr	yes	
pown	yes	
pow	no	Limited support planned
sqrt	yes	
exp	yes	
exp2	no	Planned
exp10	no	Planned
log	yes	
log2	no	Planned
log10	no	Planned
expm1	no	Planned
exp2m1	no	Planned
exp10m1	no	Planned
logp1	no	Planned
log2p1	no	Planned
log10p1	no	Planned
sin	yes	
cos	yes	
tan	no	Planned
asin	no	Planned
acos	no	Planned
atan	no	Planned
atan2	no	?
sinh	no	Planned
cosh	no	Planned
tanh	no	Planned
asinh	no	Planned
acosh	no	Planned
atanh	no	Planned
abs	no	Unclear for AA
rSqrt	no	Planned
hypot	no	Planned
compoundm1	no	Planned