interval user manual

(inter,val]

Title	interval (Interval arithmetic API for ANSI C)
Author	Nikolaos Kavvadias (C) 2009, 2010, 2011, 2012, 2013, 2014
Contact	nikos@nkavvadias.com
Website	http://www.nkavvadias.com
Release Date	29 November 2014
Version	0.2.4
Rev. history	
v0.2.4	2014-11-29 Added project logo in README.
v0.2.3	2014-11-27 Removed subsection numbers in README.
v0.2.2	2014-10-28 Added names to prototype parameters.
v0.2.1	2014-09-21 Minor style changes to README.rst.
v0.2.0	2014-09-20 Updated for github; self-contained version not depending on external files (genmacros.h, utils.c, utils.h).
v0.1.2	2009-08-11 Changes to IntervalIntersection in order to take account the case of produced empty intervals.
v0.1.1	2009-07-23 Added: IntervalIsSymmetric, IntervalMod, IntervalSet, IntervalCopy, IntervalUniverse. Changed the vmax and vmin Interval struct fields to the more formal supr (supremum) and infm (infimum).

v0.1.0	2009-07-22 Initial version. Implemented the backbone of the interval arithmetic API: INTERVAL, IntervalAdd, IntervalSub, IntervalNeg, IntervalMul, IntervalDiv, IntervalMux, IntervalAnd, IntervalIor, IntervalXor, IntervalNot, IntervalExpInteger, IntervalSqrt, IntervalAbs, IntervalMax, IntervalMin, IntervalUnion, IntervalIntersection, ValueIsInInterval, IntervalIsEmpty, IntervalIsPositive, IntervalIsNegative, ValueToInterval, IntervalBalanced, IntervalIsBalanced, IntegerBitwidthToInterval, IntervalToIntegerBitwidth, IntervalPrint.

1. Introduction

`interval` is an ANSI C implementation of a basic interval arithmetic API. The implementation of intervals is partially based on:

H. Yamashita, H. Yasuura, F.N. Eko and C. Yun,
"Variable Size Analysis and Validation of Computation Quality,"
Proceedings of the IEEE International High-Level Design Validation and Test
Workshop 2000, pp. 95--100, Berkeley, California, USA, November 8-10, 2000.

> The draft of the reference paper is available (as of 2014-Sep-20) from:
>
> - http://soc.ait.kyushu-u.ac.jp/AnnualReport/pdf/rep00/Yamashita2.pdf

2. File listing

The `interval` ADT and API code base includes the following files:

/interval	Top-level directory
AUTHORS	List of authors.
LICENSE	License argeement (Modified BSD license).
Makefile	GNU Makefile for building `test-interval.exe`.
README.rst	This file.
README.html	HTML version of README.
README.pdf	PDF version of README.
VERSION	Current version.
interval.c	C code implementing the Interval API along with some helper functions.
interval.h	C header file for the above. Also defines some arithmetic macros needed.
interval.png	PNG image for the `interval` project logo.
rst2docs.sh	Bash script for generating the HTML and PDF versions.

test-interval.c	Application code for exercising basic functionality of the implemented interval API.

3. Function reference

This section provides a quick reference of the functions used for implementing the `interval` API.

INTERVAL

```
Interval INTERVAL(int u, int v);
```

Construct an interval specified by a minimum (u) and a maximum (v) integer value. Values u and v are considered to be included in the interval.

IntervalCopy

```
Interval IntervalCopy(Interval x);
```

Return a copy of the given input interval.

IntervalEmpty

```
Interval IntervalEmpty(void);
```

Return an empty interval; interval [1,0] is produced.

IntervalUniverse

```
Interval IntervalUniverse(int bw, ArithType ztyp);
```

Returns the entire interval for a given arithmetic representation type (ztyp) and for the specified bitwidth (bw).

IntervalClamp

```
Interval IntervalClamp(Interval x, int lo, int hi);
```

Return a saturated version of the given interval for the specified lower (lo) and higher (hi) bounds.

IntervalAdd

```
Interval IntervalAdd(Interval x, Interval y);
```

Return the interval of the result of adding the intervals of two integers.

IntervalSub

```
Interval IntervalSub(Interval x, Interval y);
```

Return the interval of the result of performing subtraction on two integer intervals.

IntervalNeg

```
Interval IntervalNeg(Interval x);
```

Return a negated interval by negating the supremum and infimum fields.

IntervalMul

```
Interval IntervalMul(Interval x, Interval y, ArithType
xtyp, ArithType ytyp);
```

Return the interval of the result of performing multiplication on two integer intervals. The result is not truncated. xtyp, ytyp provide the arithmetic representation type for x and y, respectively.

IntervalDiv

```
Interval IntervalDiv(Interval x, Interval y, ArithType
xtyp, ArithType ytyp);
```

Return the interval of the result of performing division (quotient only) between two integer intervals. xtyp, ytyp provide the arithmetic representation type for x and y, respectively.

IntervalMod

```
Interval IntervalMod(Interval x, Interval y, ArithType
xtyp);
```

Return the interval of the result of performing the modulus on two integer intervals. xtyp provides the arithmetic representation type for x.

IntervalMux

```
Interval IntervalMux(Interval x, Interval y);
```

Return the interval of the result of z = ((a) relop (b) ? (x) : (y)), where relop is a relational operator:

- "==" (muxeq),
- "!=" (muxne),
- "<" (muxlt),
- "<=" (muxle),

- ">" (muxgt),
- ">=" (muxge)

IntervalSet

```
Interval IntervalSet(Interval x, Interval y);
```

Return the interval of the result of z = x relop y, where relop is a relational operator:

- "==" (seteq),
- "!=" (setne),
- "<" (setlt),
- "<=" (setle),
- ">" (setgt),
- ">=" (setge)

IntervalAnd

```
Interval IntervalAnd(Interval x, Interval y);
```

Return the interval of the result of z = x AND y.

IntervalIor

```
Interval IntervalIor(Interval x, Interval y);
```

Return the interval of the result of z = x IOR y.

IntervalXor

```
Interval IntervalXor(Interval x, Interval y);
```

Return the interval of the result of z = x XOR y.

IntervalNot

```
Interval IntervalNot(Interval x);
```

Return the interval of the result of z = NOT x.

IntervalExpInteger

```
Interval IntervalExpInteger(Interval x, int n);
```

Return the interval of the result of z = x ** n (n-th integer power of x). n is an integer and its interval representation is [n,n].

IntervalSqrt

```
Interval IntervalSqrt(Interval x);
```

Return the interval of the result of z = sqrt(x).

IntervalAbs

```
Interval IntervalAbs(Interval x);
```

Return the interval of the result of computing the absolute value of interval x: z = abs(x).

IntervalMax

```
Interval IntervalMax(Interval x, Interval y);
```

Return the interval of the result of computing the maximum value of intervals x and y: z = max(x, y).

IntervalMin

```
Interval IntervalMin(Interval x, Interval y);
```

Return the interval of the result of computing the minimum value of intervals x and y: z = min(x, y).

IntervalUnion

```
Interval IntervalUnion(Interval x, Interval y);
```

Return the union (actually the so-called "interval hull" which produces a contiguous interval) of intervals x and y. The union operator formally produces two distinct intervals.

IntervalIntersection

```
Interval IntervalIntersection(Interval x, Interval y);
```

Return the intersection of intervals x and y. In case the intersection of x and y is the empty interval, the [1,0] interval (the default empty interval) is returned.

ValueIsInInterval

```
int ValueIsInInterval(Interval x, int v);
```

Query whether the given value v is in interval x or not. Returns 1 if v is in x; 0 otherwise.

IntervalIsEmpty

```
int IntervalIsEmpty(Interval x);
```

Query whether the given interval is an empty set (i.e. containing no values). Returns 1 if the interval x is empty; 0 otherwise.

IntervalIsPositive

```
int IntervalIsPositive(Interval x);
```

Query whether the given interval is strictly positive (i.e. lies in the domain of positive integers). The interval may contain integer ZERO. Returns 1 if the interval x is positive; 0 otherwise.

IntervalIsNegative

```
int IntervalIsNegative(Interval x);
```

Query whether the given interval is strictly negative (i.e. lies in the domain of negative integers). The interval may contain integer ZERO. Returns 1 if the interval x is negative; 0 otherwise.

ValueToInterval

```
Interval ValueToInterval(int v);
```

Convert a given integer value v to a degenerate interval of the form [v,v]. Returns the computed interval.

IntervalBalanced

```
Interval IntervalBalanced(Interval x, ArithType xtyp);
```

Given an "unbalanced" interval (of the form `[m,n]`, where `m!=n` and `m,n>0` or `m<0<=n` and `|m|=n+1`), it is converted to a "balanced" interval of the form `[0,2^n-1]` for unsigned or `[-2^(n-1),2^(n-1)+1]` for signed integer arithmetic. xtyp provides the arithmetic type for the assumed integer arithmetic.

IntervalIsBalanced

```
int IntervalIsBalanced(Interval x, ArithType xtyp);
```

Query whether the given interval is balanced, i.e. `[0,2^n-1]` for unsigned or `[-2^(n-1),2^(n-1)+1]` for signed integer arithmetic. Returns 1 if the interval x is balanced; 0 otherwise.

IntervalIsSymmetric

```
int IntervalIsSymmetric(Interval x);
```

Query whether the given interval is symmetric, i.e. [-n,n] for any given arithmetic (even a non fixed-point one). Returns 1 if the interval x is symmetric; 0 otherwise.

NOTE: For non-exact arithmetic representations, the comparison operation should be carefully designed.

IntegerBitwidthToInterval

```
Interval IntegerBitwidthToInterval(int n, ArithType
xtyp);
```

Convert the bitwidth of a signed (2's complement) or unsigned integer number to the corresponding interval. A bitwidth of n-bits would be converted to [0,2**n-1] for an unsigned integer or [-2**(n-1),2**(n-1)-1] for a signed integer. xtyp provides the arithmetic type for the assumed integer.

IntervalToIntegerBitwidth

```
int IntervalToIntegerBitwidth(Interval x, ArithType
xtyp);
```

Convert the given interval to the corresponding minimum bitwidth necessary for the representation of signed (2's complement) or unsigned integers. xtyp provides the arithmetic type for the assumed integer representation.

IntervalPrint

```
void IntervalPrint(FILE *outfile, Interval x);
```

Print the specified interval to outfile.

4. Usage

The implementation of the interval API can be used in context of a provided test application, named `test-interval.c`. The Makefile can be used for building this application as follows:

```
$ cd interval
$ make clean ; make
```

To run the application do the following:

```
$ ./test-interval.exe
```

Executing the application will produce a stream of diagnostic messages to standard output.

5. Prerequisites

- Standard UNIX-based tools (tested with gcc-4.6.2 on MinGW/x86 and gcc-4.8.2 on Cygwin/x86/Windows 7)

  - make

  On Windows (e.g. Windows 7, 64-bit), MinGW (http://www.mingw.org) or Cygwin (http://sources.redhat.com/cygwin) are suggested.

  The sources should be able to compile without any messages on any recent Linux distribution.