

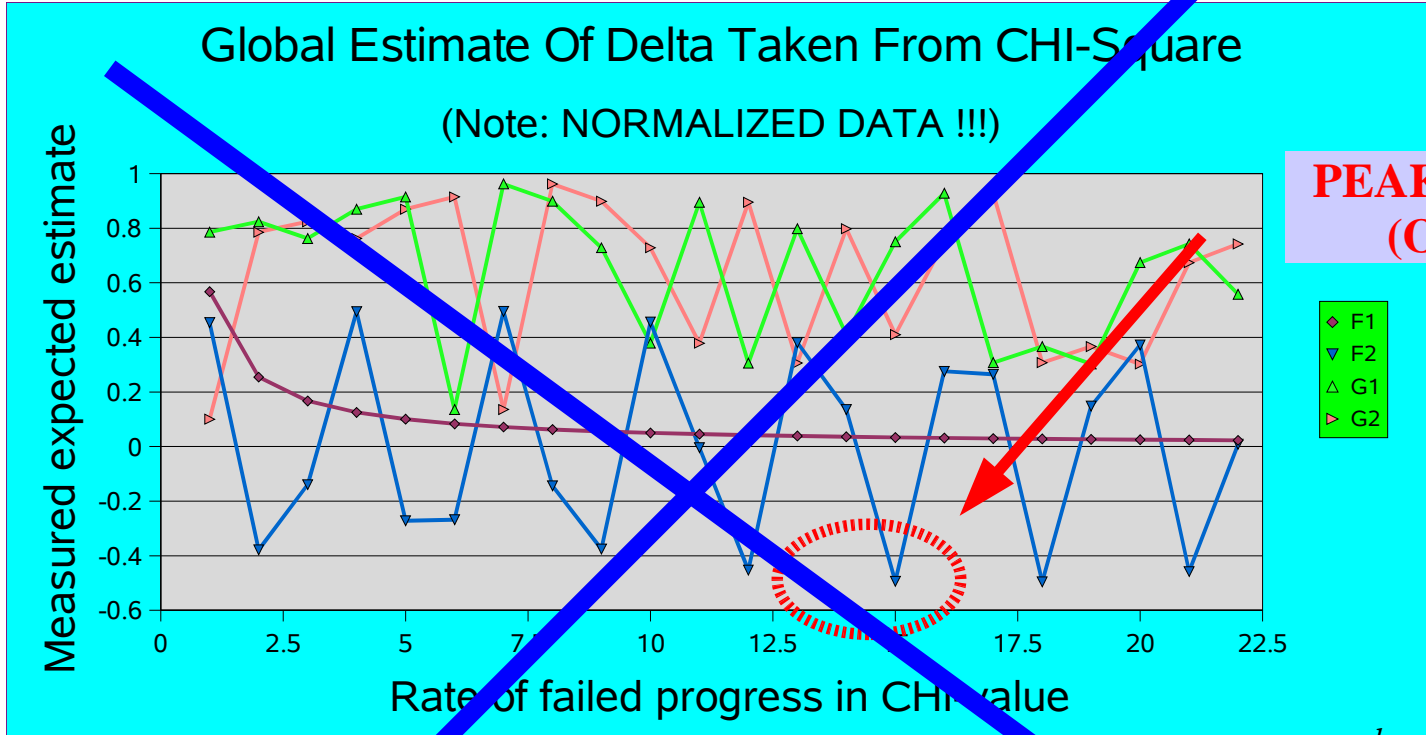
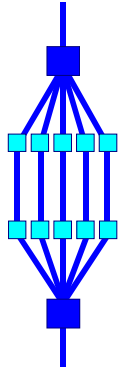
Interval Arithmetic Support In The Sun Fortran Compiler

Ruud van der Pas
Scalable Systems Group
Sun Microsystems

Bill Walster
Sun Microsystems Laboratories
Sun Microsystems

Reliable Engineering Computing Workshop
September 15-17, 2004
Savannah, Georgia, USA

This Is Not A Scientific Talk



PEAK EXPECTED ?!
(OF COURSE)



```

nextTLP[0]->previous = all_list_t_bottom[0];
all_list_t_bottom[0] = nextTLP[0];

tree_top[0]->left = tree_top[0]->right = NULL; /*initialize search tree*/
for (l=0; l<NO_IDS; l++) tree_top[0]->conf[l] = 0;
NO_FREE[0] = BLOCK_LENGTH-1;
where_to_append[0] = tree_top[0];
where_to_append[0] ++;
no_to_do[0] = 1;
no_done[0] = 0;
new_gstate_found[0] = 0;
    
```

$$G2(X, Y) = \prod_{j=0}^k Z(h, j) = \sum_{i=0}^k D(i, h) * \prod F(j)$$

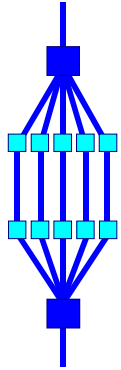
```

for (i=1; i<TWO_TO_THE_H; i++)
    
```

```

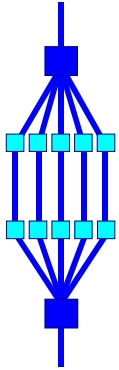
tree_top[i] = malloc((size_t)(BLOCK_LENGTH*treesize));
    
```

Outline



- *Introduction*
- *Interval-Specific Operators and Intrinsic Functions*
- *Quality Of Implementation Opportunities*
- *Conclusions*

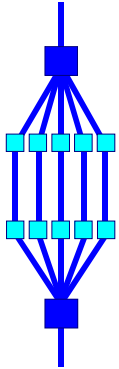
Good News For Intervals !



- *Sun has been awarded the DARPA HPCS contract*
 - *HPCS = High Productivity Computing Systems*
 - *HPCS Goal: Build a Peta-scale system*
 - *This is phase 2 of a 3-tier project*
 - *Three vendors selected (IBM, Cray and Sun)*
- *One key element in Sun's proposal is the usage of*

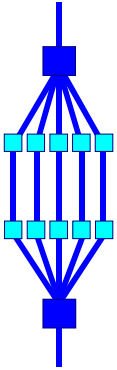
[Inter , vals]


Introduction



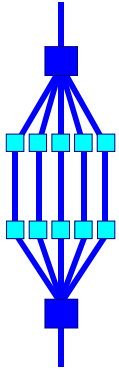
- *This presentation is joint work with Bill Walster (Sun Microsystems)*
 - *Without his relentless enthusiasm there would not have been an interval compiler from Sun*
 - *An extensive interview with Bill can be found at <http://www.sun.com/presents/minds/2004-0527>*
- *The Sun Fortran and C++ compilers support intervals since 2000 (available on SPARC based systems)*
 - *Fortran: native data type*
 - *C++: class library*
- *In this talk we would like to give an overview of the interval features supported in the Sun Fortran compiler*

Pointers To More Information/1



- **Download the compilers (Sun Studio 8)**
 - **URL is <http://developers.sun.com/prodtech/cc>**
 - **Can use a free “try and buy” license and/or**
 - **Interval support (Fortran and C++) included**
 - ✓ **Use `-xia` option to activate**
- **Or Use the large Sun systems at DTU !**
 - **Interval community can get an account; contacts:**
 - ✓ **Kaj Madsen (km@imm.dtu.dk)**
 - ✓ **Henrik Madsen (hm@imm.dtu.dk)**
 - ✓ **Bernd Damman (bd@imm.dtu.dk)** 
 - ✓ **Per Grove Thomsen (pgt@imm.dtu.dk)**

Pointers To More Information/2



□ Documentation

- *Fortran Interval Arithmetic Programming Reference*

✓ *<http://docs.sun.com/db/doc/817-5076>*

- *C++ Interval Arithmetic Programming Reference*

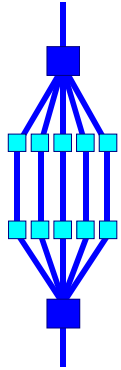
✓ *<http://docs.sun.com/db/doc/817-5077>*

- ## □ *More information plus code examples can be downloaded from <http://developers.sun.com>*

- *Go to the “Compiler Collection” portal*

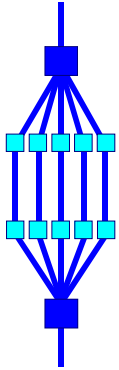
- ## □ *Another useful web site (on numerical computations):*

- *http://developers.sun.com/prodtech/cc/numerics_in dex.html*



Interval-Specific Operators and Intrinsic Functions

Basic Arithmetic Operations



Assume that $[a,b]$ and $[c,d]$ are intervals

For a basic operator "op" in $\{+,-,*,/\}$ we can then define:

$$[a,b] \text{ "op" } [c,d] \supseteq \{x \text{ "op" } y \mid x \in [a,b] \text{ and } y \in [c,d]\}$$

Formulas for basic operations:

$$[a, b] + [c, d] = [a+c, b+d]$$

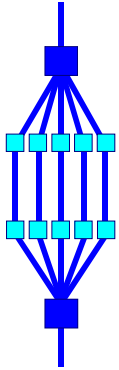
$$[a, b] - [c, d] = [a-d, b-c]$$

$$[a, b] * [c, d] = [\min(a*c, a*d, b*c, b*d), \max(a*c, a*d, b*c, b*d)]$$

$$[a, b] / [c, d] = [\min(a/c, a/d, b/c, b/d), \max(a/c, a/d, b/c, b/d)]$$

(if 0 is not included in $[c,d]$)

Integer Powers



The Dependence Problem:

$$[-1, 2] * [-1, 2] = [-2, 4]$$

The Sun Compiler will do
the right thing:

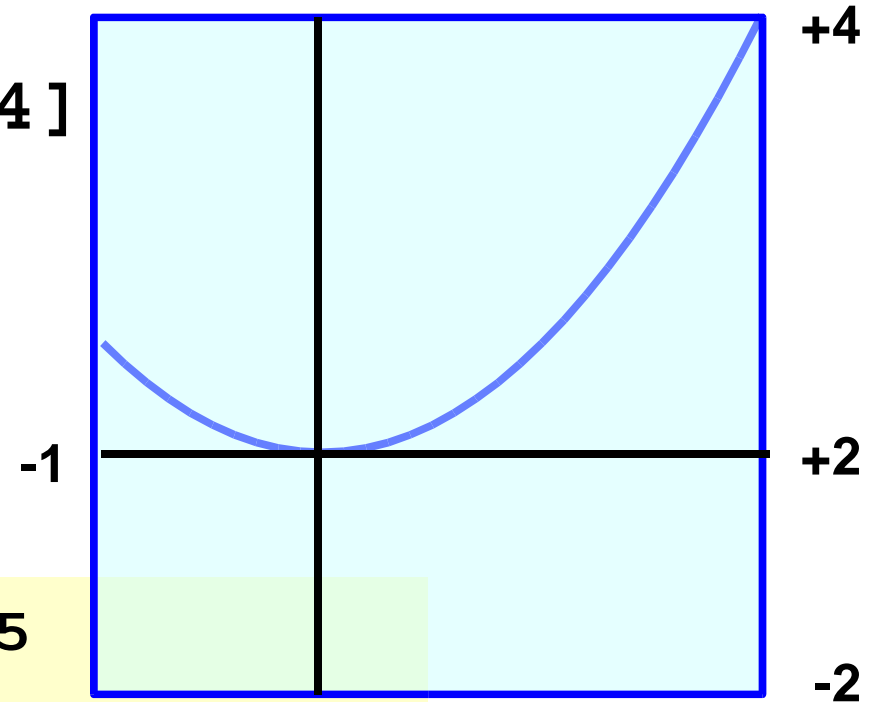
```
% f95 -o pow -xia pow.f95
```

```
% ./pow
```

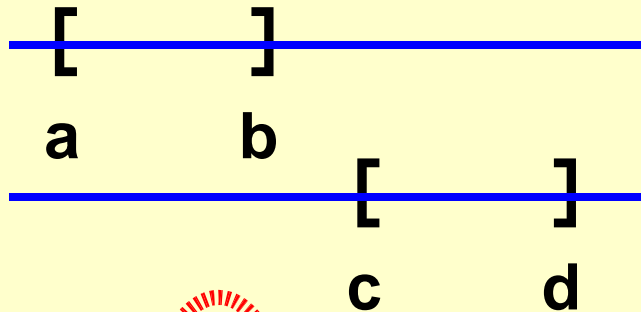
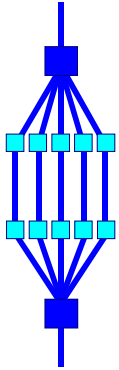
```
X      = [ -1.000000000,  2.000000000]
```

```
X*X    = [ -2.000000000,  4.000000000]
```

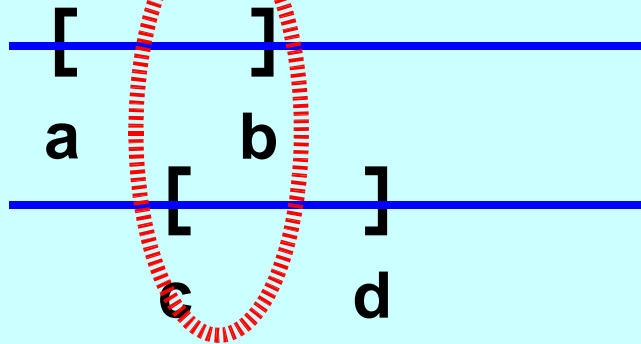
```
X**2  = [  0.000000000,  4.000000000]
```



Order Relations - What To Do ?



$[a,b]$ certainly less than $[c,d]$



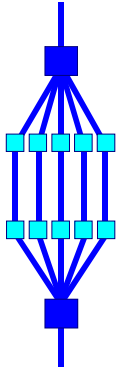
$[a,b]$ possibly less than $[c,d]$

Implementation in the Sun compiler:

One of {C, P, S}, followed by LT/LE/EQ/NE/GE/GT

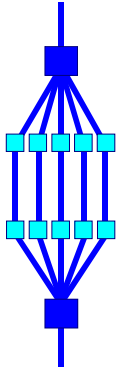
Example: A .CLT. B

Set-Theoretic Interval Operators



Name	Math. Notation	Fortran	Result Type
Interval hull	$X \underline{\cup} Y$	<code>X .IH. Y</code>	Interval
Intersection	$X \cap Y$	<code>X .IX. Y</code>	Interval
Disjoint	$X \cap Y = \emptyset$	<code>X .DJ. Y</code>	Logical
Element	$r \in Y$	<code>R .IN. Y</code>	Logical
Interior	$\underline{X} < \underline{Y}$ and $\overline{X} < \overline{Y}$	<code>X .INT. Y</code>	Logical
Proper subset	$X \subset Y$	<code>X .PSB. Y</code>	Logical
Proper superset	$X \supset Y$	<code>X .PSP. Y</code>	Logical
Subset	$X \subseteq Y$	<code>X .SB. Y</code>	Logical
Superset	$X \supseteq Y$	<code>X .SP. Y</code>	Logical

Support For Intrinsic Functions



All Fortran intrinsic functions have an interval counterpart if they either return a REAL, or accept a REAL type argument

```
% cat -n cos.f95
1 program demo
2
3 print *, 'cos (-0.5)           = ', cos(-0.5D0)
4 print *, 'cos (+0.5)           = ', cos(+0.5D0)
5 print *, 'cos [-0.5,+0.5] = ', cos([-0.5,+0.5])
6
7 stop
8 end
```

```
% f95 -o cos -xia cos.f95
```

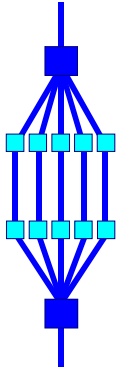
```
% ./cos
```

```
cos (-0.5)           = 0.8775825618903728
```

```
cos (+0.5)           = 0.8775825618903728
```

```
cos [-0.5,+0.5] = [0.87758256189037264, 1.0]
```

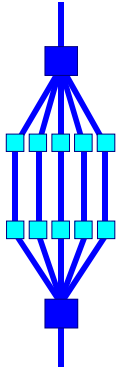
Interval Specific Intrinsic



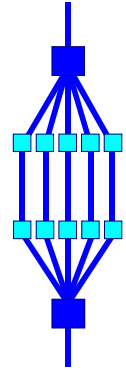
Name	Definition	Name	Result Type
Infimum	$\text{inf}([a,b]) = a$	INF	REAL
Supremum	$\text{sup}([a,b]) = b$	SUP	REAL
Width	$w([a,b]) = b-a$	WID	REAL
Midpoint	$(a+b) / 2$	MID	REAL
Magnitude	$\max(a , b)$	MAG	REAL
Mignitude	$\min(a , b)^*$	MIG	REAL
Empty Test	TRUE if empty	ISEMPTY	LOGICAL
Number Of Digits	Max. digits	NDIGITS	INTEGER

**) Returns 0 if $0 \in [a,b]$*

Input / Output

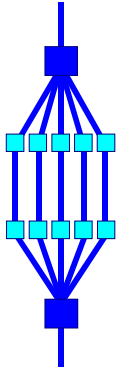


- *Square brackets (“[“and “]”) are used to delimit intervals*
 - *Example: $X = [-0.1, 0.2]$*
- *All edit descriptors that accept REAL data items also accept INTERVAL data*
- *Specific INTERVAL edit descriptors are supported as well*



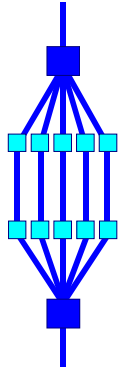
Quality Of Implementation Opportunities

Supported Features



- *A closed interval system in which all expressions (including singularities and indeterminate forms) are defined*
 - *Examples: $1/0$, x^y with $x=y=0$, operations involving $+\infty$ and/or $-\infty$*
- *Domain constraints on intrinsic functions are gracefully handled*
 - *Example: $SQRT([-1, +1]) = [0, 1]$*
- *Input / Output can be handled in different ways*
- *Context dependent literal interval constants*
- *Mixed mode expressions*

Example Code



Program Demo

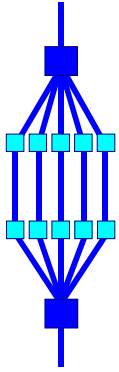
```
logical :: not_done = .true.
interval(kind=8)      :: ai, bi
write(*,*) 'Please give values for A and B'
do while ( not_done )
    read(*,*,end=9000) ai, bi

    write(*,9010) '+',ai,'+',bi,ai+bi
    write(*,9010) '-',ai,'-',bi,ai-bi
    write(*,9010) '*',ai,'*',bi,ai*bi
    write(*,9010) '/',ai,'/',bi,ai/bi
    write(*,*)
end do

9000 continue
stop

9010 format(1X,'A',1X,(A),1X,'B =',VF17.4,1X,(A), &
1X,VF17.4,' = ',VF17.4)
end
```

Example Closed Interval System



```
% f95 -xia math.f95
```

```
% ./a.out
```

Please give values for A and B

```
A + B = [-1.0000, 3.0000] + [ 1.0000, 2.0000] = [ 0.0000, 5.0000]
```

```
A - B = [-1.0000, 3.0000] - [ 1.0000, 2.0000] = [-3.0000, 2.0000]
```

```
A * B = [-1.0000, 3.0000] * [ 1.0000, 2.0000] = [-2.0000, 6.0000]
```

```
A / B = [-1.0000, 3.0000] / [ 1.0000, 2.0000] = [-1.0000, 3.0000]
```

```
A + B = [ 1.0000, 2.0000] + [-1.0000, 3.0000] = [ 0.0000, 5.0000]
```

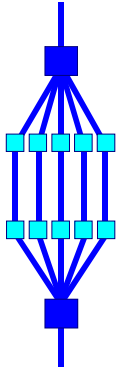
```
A - B = [ 1.0000, 2.0000] - [-1.0000, 3.0000] = [-2.0000, 3.0000]
```

```
A * B = [ 1.0000, 2.0000] * [-1.0000, 3.0000] = [-2.0000, 6.0000]
```

```
A / B = [ 1.0000, 2.0000] / [-1.0000, 3.0000] = [ -Inf, Inf]
```

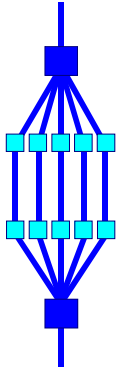


Summary



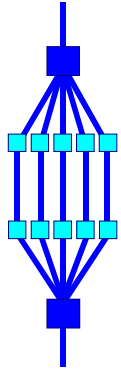
- ❑ *The Sun Fortran and C++ compilers support Interval Arithmetic*
- ❑ *The regular Basic Arithmetic Operations, intrinsic functions and logical operations have been extended to intervals*
- ❑ *In addition to this, several quality of implementation features are supported:*
 - *Closed interval system, domain constraints on intrinsic functions, input/output, ontext dependent literal interval constants, etc.*
- ❑ *We believe that this provides for a production quality interval compiler*

We Need Your Help



For Sun to continue interval support, we need you to use our compilers and give feedback

***Feel free to send me an email:
ruud.vanderpas@sun.com***



Thank You !