



## Formal proofs of numerical programs

Nguyen Thi Minh Tuyen, Sylvie Boldo, Claude Marché

► **To cite this version:**

Nguyen Thi Minh Tuyen, Sylvie Boldo, Claude Marché. Formal proofs of numerical programs. Forum Digitéo, Oct 2010, Palaiseau, France. <inria-00536135>

**HAL Id: inria-00536135**

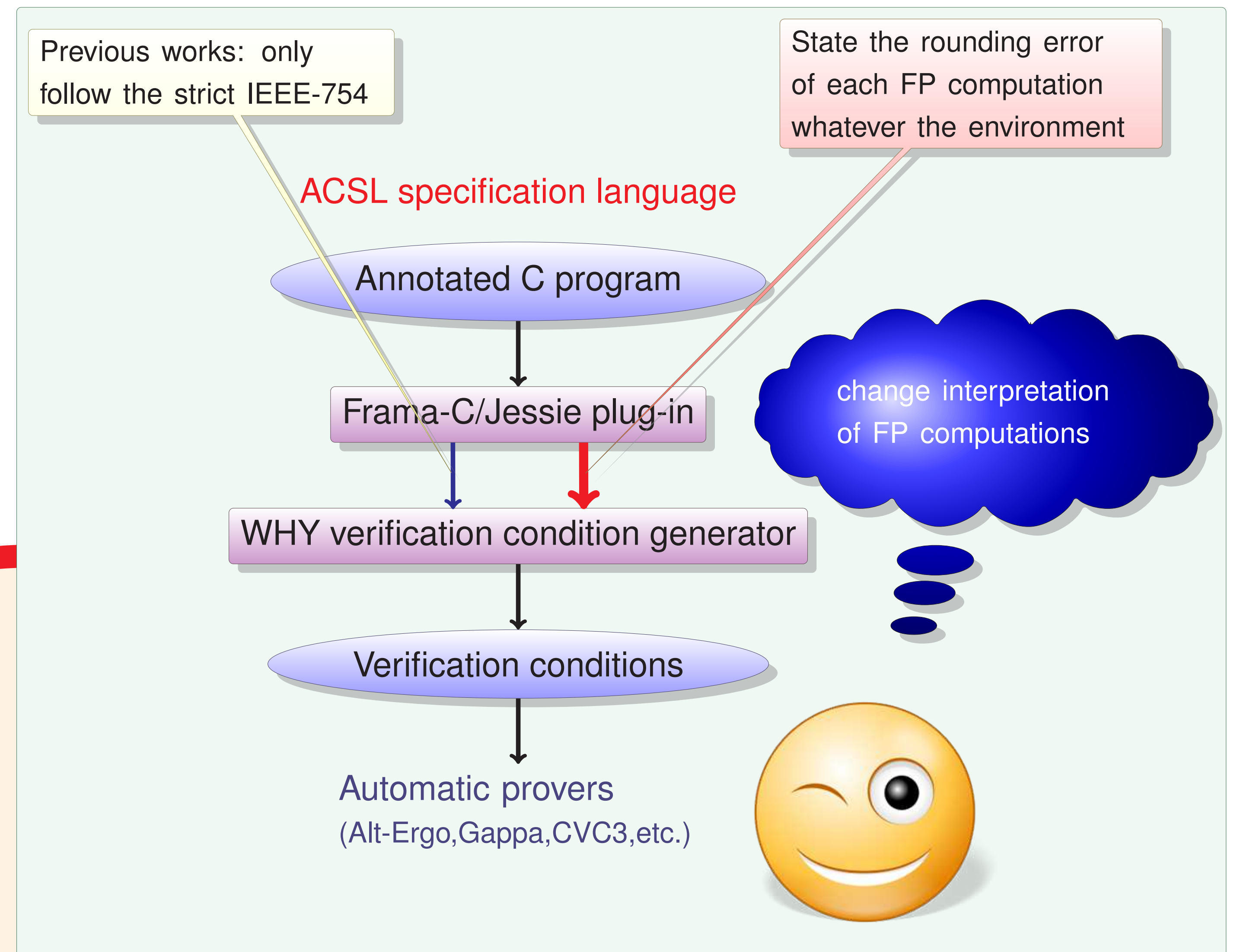
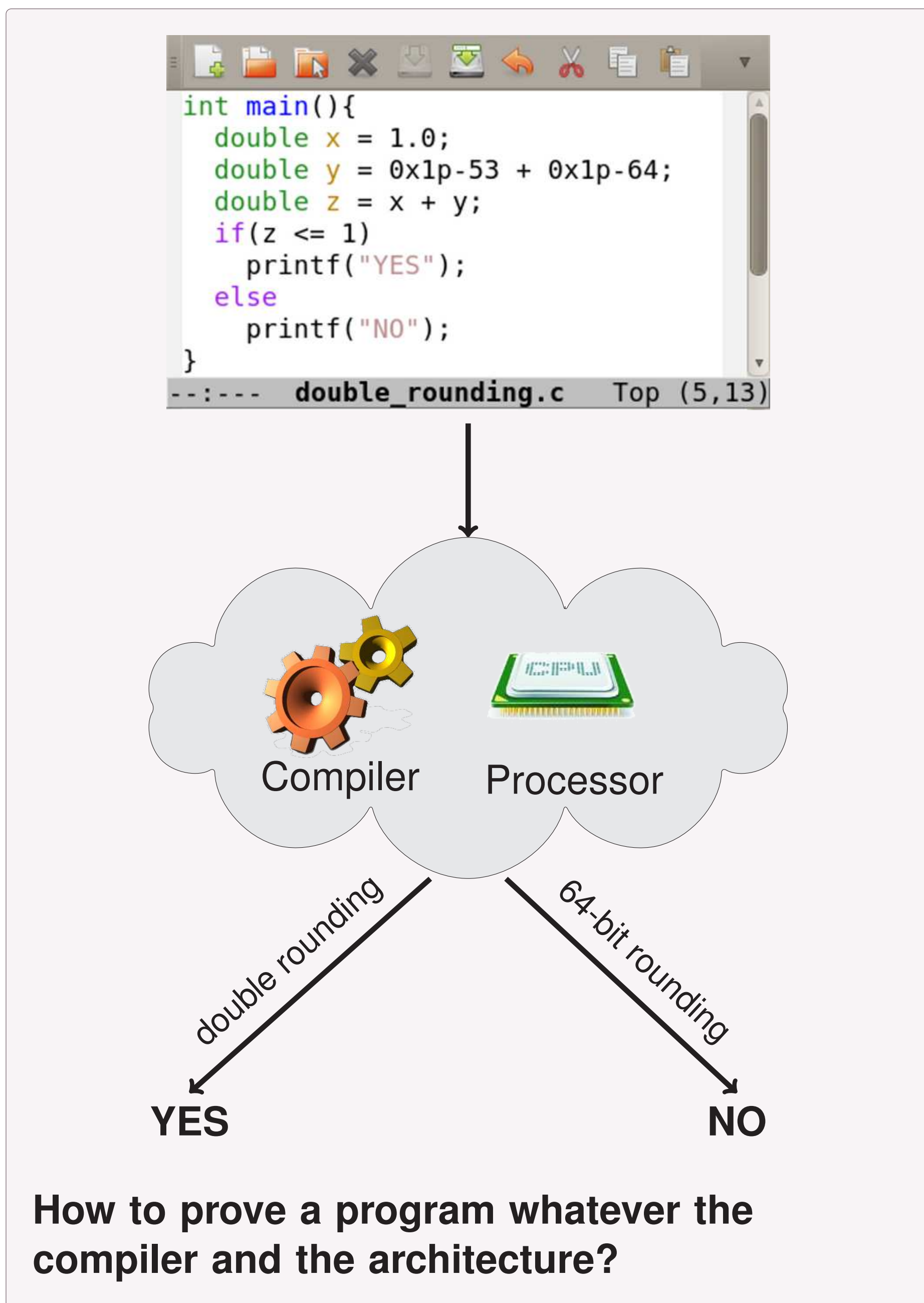
**<https://hal.inria.fr/inria-00536135>**

Submitted on 15 Nov 2010

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

## FORMAL PROOFS OF NUMERICAL PROGRAMS



### Theorem

Let  $\varepsilon' = 2051 \cdot 2^{-60}$  and  $\varepsilon = 2050 \times 2^{-64}$ . Let  $\eta' = 2049 \times 2^{-1082}$  and  $\eta = 2049 \times 2^{-1086}$ . If we define each operation result as any real such that

$$\begin{aligned}
 |x \oplus y - (x + y)| &\leq \varepsilon' \cdot (|x| + |y|) + \eta' \\
 |x \ominus y - (x - y)| &\leq \varepsilon' \cdot (|x| + |y|) + \eta' \\
 |x \otimes y - (x * y)| &\leq \varepsilon \cdot |x * y| + \eta \\
 |x \oslash y - (x / y)| &\leq \varepsilon \cdot |x / y| + \eta
 \end{aligned}$$

and if we are able to deduce a property (such as a rounding error), then this property holds whatever the architecture and the compiler optimizations among commutativity, addition/subtraction associativity (for less than 16 additions), use of FMA, use of extended registers, expression factorization and unfactorization.

```

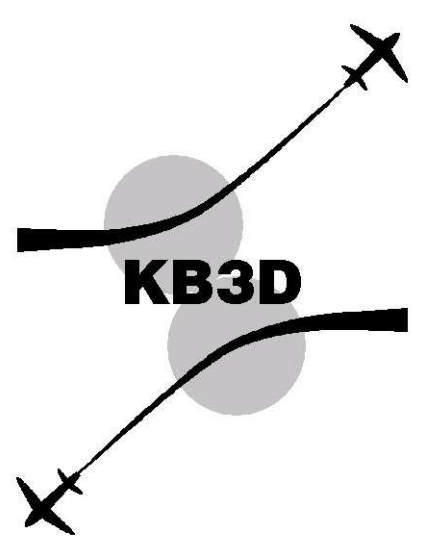
File Edit Options Buffers Tools C Help
Logic integer l_sign(real x) = (x >= 0.0) ? 1 :-1;

/* requires e1 <= x - \exact(x) <= e2;
   @ ensures \abs(\result) <= 1 &&
   @ (\result != 0 ==> \result == l_sign(\exact(x)));
   */
int sign(double x, double e1, double e2) {
  if (x > e2) return 1;
  if (x < e1) return -1;
  return 0;
}

/* requires
   @ sx == \exact(sx) && sy == \exact(sy) &&
   @ vx == \exact(vx) && vy == \exact(vy) &&
   @ \abs(sx) <= 100.0 && \abs(sy) <= 100.0 &&
   @ \abs(vx) <= 1.0 && \abs(vy) <= 1.0;
   @ ensures
   @ \result != 0
   @ ==> \result == l_sign(\exact(sx)*\exact(vx) + \exact(sy)*\exact(vy))
   @ * l_sign(\exact(sx)*\exact(vy) - \exact(sy)*\exact(vx));
   */
int eps_line(double sx, double sy, double vx, double vy) {
  int s1 = sign(sx*vx + sy*vy, -0x1.aap-42, 0x1.aap-42);
  int s2 = sign(sx*vy - sy*vx, -0x1.aap-42, 0x1.aap-42);
  return s1*s2;
}
    
```

Proof obligations	Alt-Ergo	CVC3	Gappa	Statistics
Function eps_line	0.91	2.2 (SS)	0.13.0	1/1
Default behavior	✓	✓	✓	1/1
1. precondition	✓	✓	✓	13/13
Function eps_line	✓	✓	✓	13/13
Safety	✓	✓	✓	13/13
1. check FP overflow	✓	✓	✓	13/13
2. check FP overflow	✓	✓	✓	13/13
3. check FP overflow	✓	✓	✓	13/13
4. check FP overflow	✓	✓	✓	13/13
5. check FP overflow	✓	✓	✓	13/13
6. precondition for user	✓	✓	✓	13/13
7. precondition for user	✓	✓	✓	13/13
8. check FP overflow	✓	✓	✓	13/13
9. check FP overflow	✓	✓	✓	13/13
10. check FP overflow	✓	✓	✓	13/13
11. check FP overflow	✓	✓	✓	13/13
12. check FP overflow	✓	✓	✓	13/13
13. precondition for use	✓	✓	✓	13/13
Function sign	✓	✓	✓	6/6
Default behavior	✓	✓	✓	6/6
1. precondition	✓	✓	✓	6/6
2. precondition	✓	✓	✓	6/6
3. precondition	✓	✓	✓	6/6
4. precondition	✓	✓	✓	6/6
5. precondition	✓	✓	✓	6/6
6. precondition	✓	✓	✓	6/6

Case study  
Part of KB3D NASA  
<http://research.nianet.org/fm-at-nia/KB3D/>



### Future work

- Look into multiplication/division reordering
- Look into assembly
- know the order of the operations
- know the precision used for each operation
- know if the architecture supports FMA or not