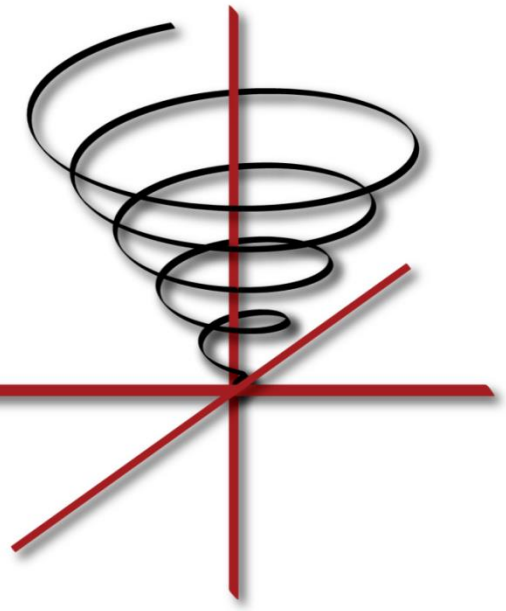
High Assurance Spiral:
Co-Synthesizing Proof and Implementation from High-Level Specification



Franz Franchetti
Carnegie Mellon University

In collaboration with
K. Ghorbal, J. Johnson, S. Mitsch, A. Platzer, J.D. Quesel, R. Veras

Sponsored by the DARPA HACMS Program under agreement FA8750-12-2-0291

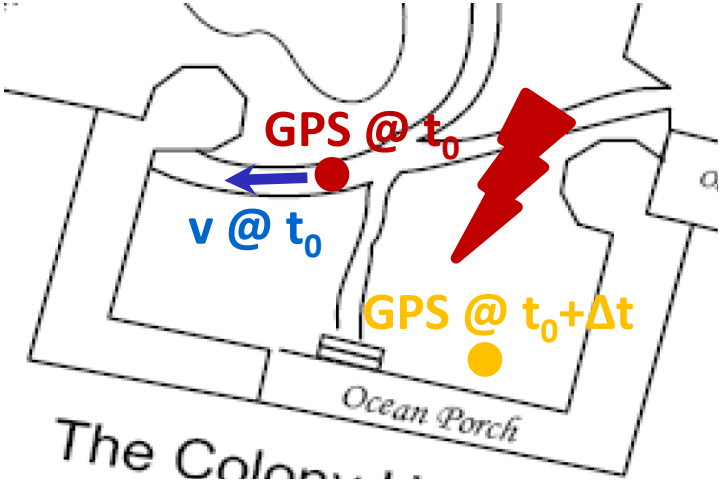
Motivation: Model-Based High Assurance

Multi-sensor ground vehicles

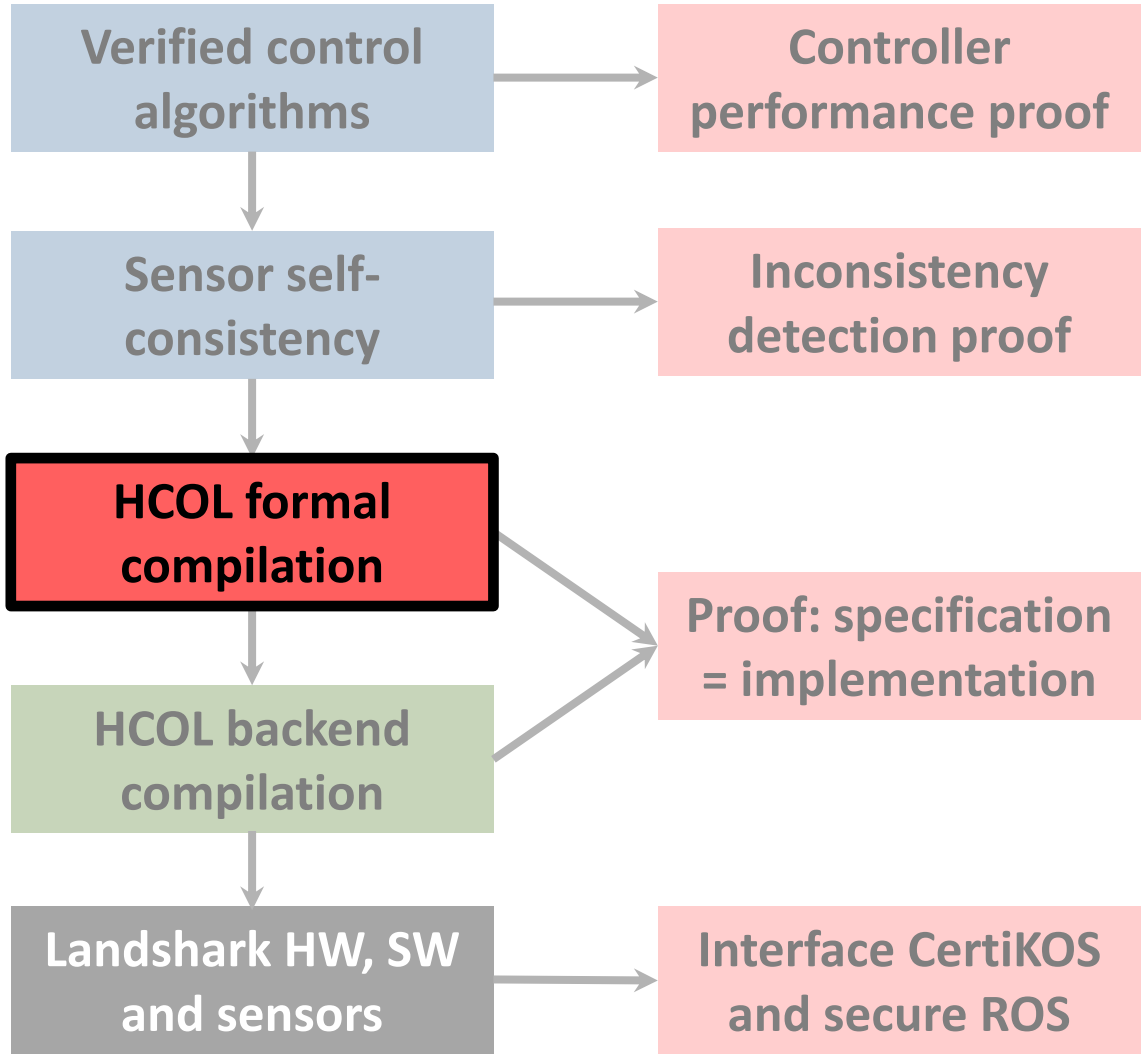
- **Multiple sensors:** GPS, compass, accelerometer, IMU, etc.
- **Control:** waypoints, joystick vector
- **Vehicle model:** laws of physics, vehicle state
- **Map data:** Terrain, possible paths, obstacles

Assurance Through Consistency

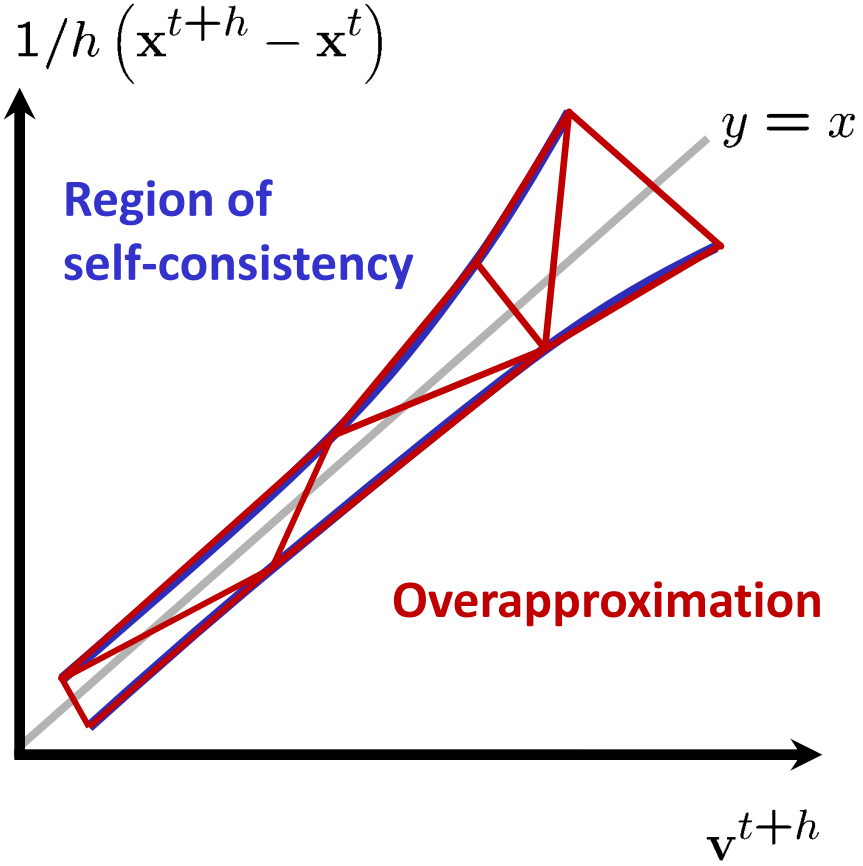
- Model-based consistency checks
- Model vs. vehicle state
- Map-based path validation
- Exception signal if inconsistency threshold is exceeded



High Assurance Code/Proof Co-Synthesis



Detection Through Feasible Region of State



Self-consistency equation

$$\mathcal{F}:\begin{pmatrix}\mathbf{x}^{t}\\ \mathbf{v}^{t}\\ \mathbf{x}^{t+h}\\ \mathbf{v}^{t+h}\end{pmatrix}\mapsto\begin{pmatrix}1/h\left(\mathbf{x}^{t+h}-\mathbf{x}^{t}\right)\\ \mathbf{v}^{t+h}\end{pmatrix}$$

Inside a polyhedra

$$A_i\mathcal{F}(\vec{x})-b_i\preceq\vec{0}$$

Test: attack-free, if $\mathcal{F}\left(\mathbf{s}^{t}\oplus\mathbf{s}^{t+h}\right)\in\bigcup_i\mathcal{P}_i$

Rule-Based Formal Compilation Flow

HCOL specification

$$\mathrm{ForAny}_{i=0}^{k-1}\left(\mathrm{InsidePoly}_{m,n}(\mathbf{A}_i, \mathbf{b}_i, .)\right)$$

Expansion + backtracking

HCOL (dataflow) expression

$$\mathrm{Reduction}_{n,(a,b)\mapsto a\vee b} \circ \left[\cdot\right]_{\ell=0}^{k-1}\Big(\mathrm{Reduction}_{n,(a,b)\mapsto a\wedge b} \circ \mathrm{Pointwise}_{n,x_i\mapsto x_i\leq 0} \circ \mathrm{Pointwise}_{n,x_i\mapsto x_i-b_i} \circ \left[\cdot\right]_{i=0}^{m-1}\left(\mathrm{Reduction}_{n,(a,b)\mapsto a+b} \circ \mathrm{Pointwise}_{n,x_j\mapsto a_{i,j}x_j}\right)\Big)$$

Recursive descent

Σ-HCOL (loop) expression

$$\mathrm{Reduction}_{n,(a,b)\mapsto a\vee b} \circ \sum_{\ell=0}^{k-1}\!\!\cup\Big(\mathrm{e}_\ell^k \circ \mathrm{Reduction}_{n,(a,b)\mapsto a\wedge b} \circ \mathrm{Pointwise}_{n,x_i\mapsto x_i\leq 0} \circ \mathrm{Pointwise}_{n,x_i\mapsto x_i-b_i} \circ \sum_{i=0}^{m-1}\!\!\cup\Big(\mathrm{e}_i^m \circ \mathrm{Reduction}_{n,(a,b)\mapsto a+b} \circ \sum_{j=0}^{n-1}\!\!\cup\Big(\mathrm{e}_j^n \circ \mathrm{Atomic}_{x_j\mapsto a_{i,j}x_j} \circ (\mathrm{e}_j^n)^\top\Big)\Big)\Big)$$

Confluent term rewriting

Optimized Σ-HCOL expression

$$\sum_{\ell=0}^{k-1}\!\!\vee\Big(\sum_{i=0}^{m-1}\!\!\wedge\Big(\mathrm{Atomic}_{x\mapsto(x-b_i)\leq 0} \sum_{j=0}^{n-1}\!\!+\Big(\mathrm{Atomic}_{x\mapsto a_{i,j}x} \circ (\mathrm{e}_j^i)\top\Big)\Big)\Big)$$

Recursive descent

Abstract code

```
func(TVoid, "transform", [ Y, X ],
       data(D1, V([ V([ V([ V(1.0), V(2.0) ]), V([ V(3.0), V(4.0) ]), V([ V(5.0), V(
          data(D2, V([ V([ V(1.0), V(2.0), V(3.0) ]), V([ V(1.0), V(2.0), V(3.0) ])
             chain(
                assign(nth(Y, V(0)), V(false)),
                loopc(i1, [ 0 .. 1 ], neq(nth(Y, V(0)), V(true)),
                   decl([ s8 ],
                      chain(
                         assign(s8, V(true)),
                         loopc(i19, [ 0 .. 2 ], neq(s8, V(false)),
                            decl([ s10, s11, s12, s13, s14, s9 ],
```

extends [PLDI 2005, DSLWC 2009]

HCOL: Hybrid Control Operator Language

Sensor values and model-based predictions

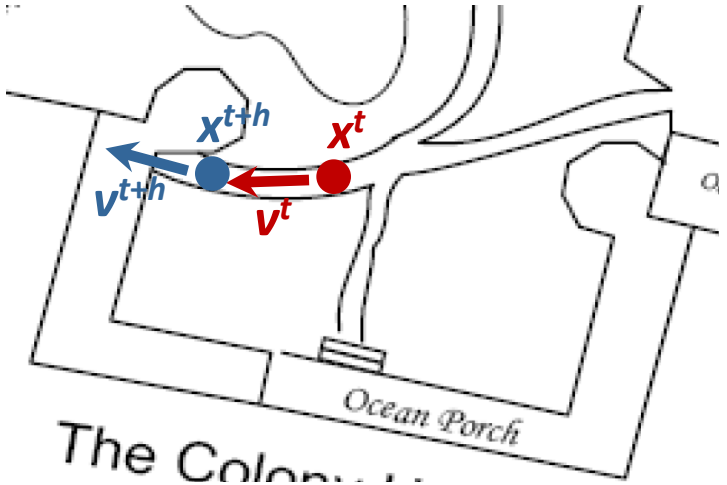
Euler step: x^{t+h}

$$\mathbf{x}^{t+h} \approx \left[\mathrm{I}_3 \,|h\,\mathrm{I}_3\right](\mathbf{x}^t \oplus \mathbf{v}^{t+h})$$

Numerical differentiation: v^{t+h}

$$\mathbf{v}^{t+h} \approx 1/h\left[\mathrm{I}_3 \,|-\mathrm{I}_3\right](\mathbf{x}^{t+h} \oplus \mathbf{x}^t)$$

I_3: 3 x 3 identity matrix
time step = matrix-vector product



Assurance through guaranteed controller input and output

- Declarative representation of physics, data and control algorithms
- Enables rule-based software synthesis and variant generation, verification and proof co-synthesis
- Extends Spiral's OL and SPL languages into the control domain

HCOL: Control Operator Examples

Time step residue: Disagreement between model and sensors

$$\mathbf{r}^{t+h} = \mathbf{R} \cdot (\mathbf{x}^t \oplus \mathbf{v}^t \oplus \mathbf{x}^{t+h} \oplus \mathbf{v}^{t+h})$$

$$\mathbf{R} = \begin{bmatrix} -\mathrm{I}_3 & h\,\mathrm{I}_3 & \mathrm{I}_3 & 0_3 \\ 1/h\,\mathrm{I}_3 & 0_3 & 1/h\,\mathrm{I}_3 & \mathrm{I}_3 \end{bmatrix}$$

Error operator: L_2 norm of time step residue

$$\mathsf{E}_h : (\mathbf{s}^t, \mathbf{s}^{t+h}) \mapsto (\mathbf{s}^t \oplus \mathbf{s}^{t+h})^\top \left(\mathbf{R}^\top \mathbf{R}\right) (\mathbf{s}^t \oplus \mathbf{s}^{t+h})$$

$$\mathbf{s}^t = (\mathbf{x}^t \oplus \mathbf{v}^t), \quad \mathbf{s}^{t+h} = (\mathbf{x}^{t+h} \oplus \mathbf{v}^{t+h})$$

PID controller: Control velocity at set point v_0

$$\begin{pmatrix} \mathbf{u}^t \\ \mathbf{s}^t \end{pmatrix} = \left(\begin{bmatrix} k_p\,\mathrm{I}_e & k_i\,\mathrm{I}_3 & k_d/h\,\mathrm{I}_3 \end{bmatrix} \begin{bmatrix} \mathrm{I}_3 & . & . \\ \mathrm{I}_3 & . & \mathrm{I}_3 \\ \mathrm{I}_3 & -\mathrm{I}_3 & . \end{bmatrix} \begin{bmatrix} \mathrm{I}_3 & -\mathrm{I}_3 & . \\ . & . & \mathrm{I}_6 \end{bmatrix} \right) \oplus \begin{bmatrix} \mathrm{I}_3 & -\mathrm{I}_3 & . \\ \mathrm{I}_3 & -\mathrm{I}_3 & \mathrm{I}_3 \end{bmatrix} (\mathbf{v}_0 \oplus \mathbf{v}^t \oplus \mathbf{s}^{t-h})$$

$$\mathbf{e}^t = \mathbf{v}^0 - \mathbf{v}^t, \quad \mathbf{s}^t = \mathbf{e}^t \oplus \sum_{i=0}^{n-1} \mathbf{e}^{ih}$$

Usual PID controller definition: $\mathbf{u}^t = k_p \mathbf{e}^t + k_i \sum_{i=0}^{n-1} \mathbf{e}^{ih} + k_d \dfrac{\mathbf{e}^t - \mathbf{e}^{t-h}}{h}$

Rule-Based Code Synthesis

High Level Rules: Transformations within high level abstraction

$$\mathrm{I}_n \rightarrow \sum_{i=0}^{n-1} \mathrm{e}_i^n \, \mathrm{I}_1 (\mathrm{e}_i^n)^\top$$

$$\left[\sum_{i=0}^{n-1} \mathsf{S}_i \, A_i \, \mathsf{G}_i \,|\, \sum_{i=0}^{n-1} \mathsf{S}_i \, B_i \, \mathsf{G}_i \right] \rightarrow \sum_{i=0}^{n-1} \mathsf{S}_i \left(\left[A_i | B_i \right] \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) \mathsf{G}_i$$

with $\mathrm{e}_i^{1 \times n} = [0, \ldots, 0, 1, 0, \ldots, 0]$

Code generation rules: Translate high level abstraction into code

$$\begin{aligned}
\mathrm{Code}\left(y = (A\,B)x\right) &\rightarrow \left\{ \mathtt{Decl}(t), \mathrm{Code}\left(t = Bx\right), \mathrm{Code}\left(y = At\right) \right\} \\
\mathrm{Code}\left(y = \left(\sum_{i=0}^{n-1} A_i \right) x \right) &\rightarrow \left\{ y := \vec{0}, \mathtt{for}(i = 0..n-1)\ \mathrm{Code}\left(y \mathrel{+}= A_i x\right) \right\} \\
\mathrm{Code}\left(y = \mathrm{e}_i^{1 \times n} x\right) &\rightarrow y[0] := x[i] \\
\mathrm{Code}\left(y = \mathrm{e}_i^{n \times 1} x\right) &\rightarrow \left\{ y = \vec{0}, y[i] := x[0] \right\}
\end{aligned}$$

Symbolic Rule Verification

- **Rule replaces left-hand side by right-hand side when preconditions match**

$$\mathrm{I}_n \rightarrow \sum_{i=0}^{n-1} \mathrm{e}_i^n \, \mathrm{I}_1 (\mathrm{e}_i^n)^\top$$

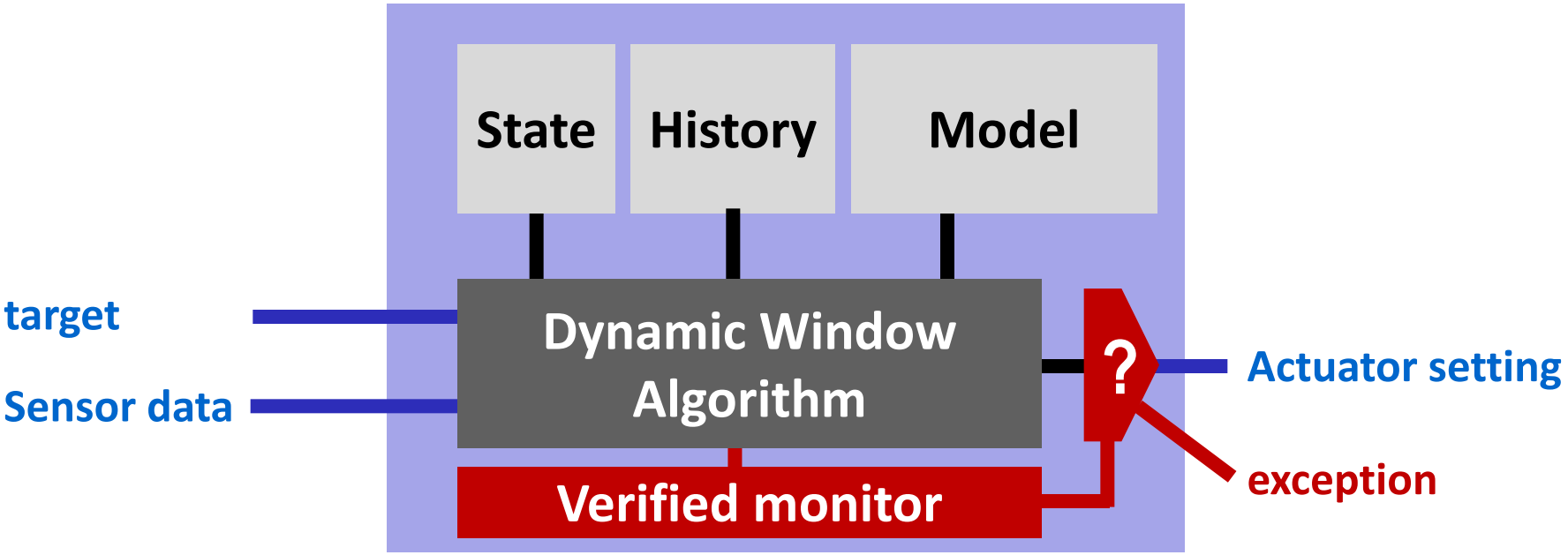
- **Test rule by symbolically evaluating expressions before and after rule application and compare result**

$$\mathrm{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad = ? \quad \sum_{i=0}^{2} \mathrm{e}_i^3 \, \mathrm{I}_1 (\mathrm{e}_i^3)^\top = [1\,0\,0][1][1\,0\,0]^\top + [0\,1\,0][1][0\,1\,0]^\top + [0\,0\,1][1][0\,0\,1]^\top$$

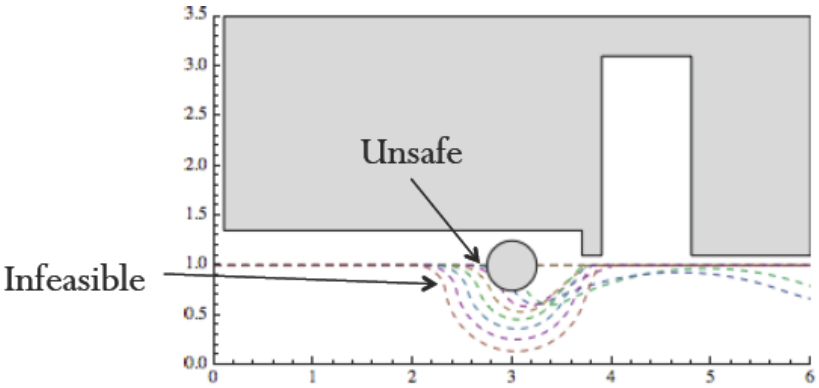
Algorithms Formalized in HA Spiral So Far

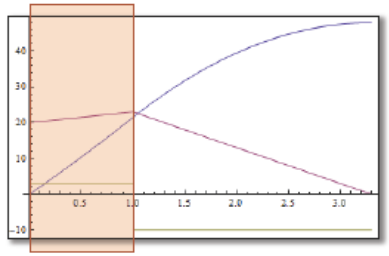
- **Dynamic Window Approach Monitor**
  Passive safety monitor, formally derived in KeYmaera
- **Set calculus: Sensor self-consistency in state space**
  Check that set of self-consistent true state values permitted by measurements is non-empty
- **Multi-timescale Z-test for redundant sensors**
  Test for zero mean of difference between multiple sensors on multiple time scales
- **Mathematical infrastructure robotics code**
  Coordinate transformations, data filtering, ODE integration

Example: Dynamic Window Safety Monitor



Dynamic Window Approach Primer





$$\|p_r - p_o\|_\infty > \frac{v_r^2}{2b} + V\frac{v_r}{b} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v_r+V)\right)$$

Algorithm Verified in KeYmaera

Theorem and proof

$$\textbf{To Prove: } \psi_{ps} \rightarrow [dw_{ps}]\left((v_r = 0) \vee \left(\|p_r - p_o\| > \frac{v_r^2}{2b} + V\frac{v_r}{b}\right)\right)$$

$$dw_{ps} \equiv (ctrl_o \;||\; ctrl_r;\; dyn)^*$$

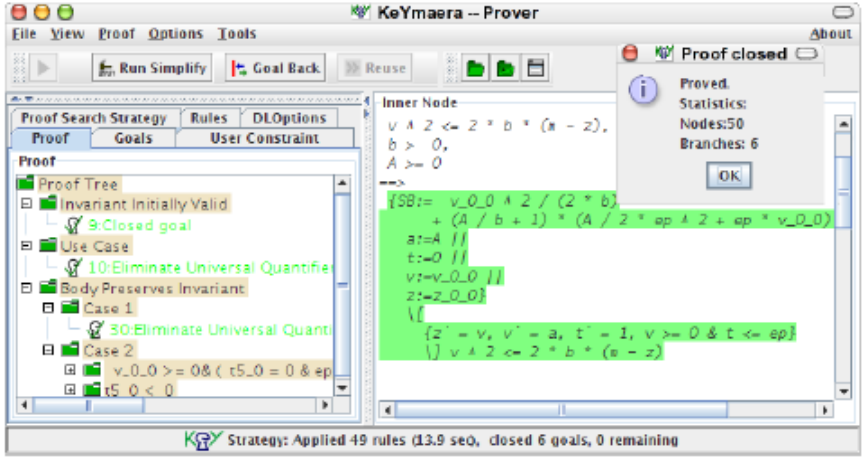
$$ctrl_o \equiv v_o = (*,*);\; ?\|v_o\| \leq V$$

$$ctrl_r \equiv (a_r := -b) \cup (?v_r = 0;\; a_r := 0;\; \omega_r := 0) \cup (a_r := *;\; ?-b \leq a_r \leq A;\; \omega_r := *;\; ? -\Omega \leq \omega_r \leq \Omega;\; p_c := (*,*);\; d_r := (*,*);\; p_o := (*,*);\; ?feasible \wedge safe)$$

$$feasible \equiv \|p_r - p_c\| > 0 \wedge \omega_r \|p_r - p_c\| = v_r \wedge d_r = \frac{(p_r - p_c)^\perp}{\|p_r - p_c\|}$$

$$safe \equiv \|p_r - p_o\|_\infty > \frac{v_r^2}{2b} + V\frac{v_r}{b} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v_r + V)\right)$$

$$dyn \equiv (t := 0;\; p_r^{x\prime} = v_r d_r^x,\; p_r^{y\prime} = v_r d_r^y,\; d_r^{x\prime} = -\omega_r d_r^y,\; d_r^{y\prime} = \omega_r d_r^x,\; p_o^{x\prime} = v_o^x,\; p_o^{y\prime} = v_o^y,\; v_r' = a_r,\; \omega_r' = \frac{a_r}{\|p_r - p_c\|},\; t' = 1 \;\&\; v_r \geq 0 \wedge t \leq \varepsilon)$$



Resulting safety monitor condition

$$\|p_r - p_o\|_\infty > \frac{v_r^2}{2b} + V\frac{v_r}{b} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v_r + V)\right)$$

	$\|p_r - p_o\|_\infty > \frac{v_r^2}{2b} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon v_r\right)$
	$v_r = 0 \vee \|p_r - p_o\|_\infty > \frac{v_r^2}{2b} + V\frac{v_r}{b} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v_r + V)\right)$
Sensor uncertainty	$\|\tilde{p}_r - p_o\|_\infty > \frac{v_r^2}{2b} + V\frac{v_r}{b} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v_r + V)\right) + U_p$
Actuator disturbance	$\|p_r - p_o\|_\infty > \frac{v_r^2}{2bU_m} + V\frac{v_r}{bU_m} + \left(\frac{A}{bU_m}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v_r + V)\right)$
Sensor failure	$\|\tilde{p}_r - p_o\|_\infty > \frac{v_r^2}{2b} + V\frac{v_r}{b} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v + V)\right) + U_p + g\Delta$
	$v_r = 0 \vee \|p_r - p_o\|_\infty > \frac{v_r^2}{2b} + \frac{V^2}{2b_o} + V\left(\frac{v_r}{b} + \tau\right) + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon(v_r + V)\right)$

Mathematical Objects in HCOL

- **Infinity norm**

$$\|.\|_\infty^n : \mathbb{R}^n \to \mathbb{R}$$
$$(x_i)_{i=0,\ldots,n-1} \mapsto \max_{i=0,\ldots,n-1} |x_i|$$

- **Chebyshev distance**

$$d_\infty^n(.,.) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$$
$$(x, y) \mapsto \|x - y\|_\infty^n$$

- **Vector subtraction**

$$(-)_n : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$$
$$(x, y) \mapsto x - y$$

Implicit Isomorphisms

$$\mathbb{R}^m \times \mathbb{R}^n \cong \mathbb{R}^{m+n}$$
$$\mathbb{R} \cong \mathbb{R}^1$$
$$\mathbb{R}^k \times (\mathbb{R}^m \times \mathbb{R}^n) \cong \mathbb{R}^k \times \mathbb{R}^m \times \mathbb{R}^n$$
$$(\mathbb{R}^k \times \mathbb{R}^m) \times \mathbb{R}^n \cong \mathbb{R}^k \times \mathbb{R}^m \times \mathbb{R}^n$$

- **Pointwise comparison**

$$(<)_n : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{Z}_2^n$$
$$\left((x_i)_{i=0,\ldots,n-1}, (y_i)_{i=0,\ldots,n-1}\right) \mapsto (x_i < y_i)_{i=0,\ldots,n-1}$$

HCOL Basic Operators

$$\text{Pointwise}_{n,f_i} : \mathbb{R}^n \to \mathbb{R}^n$$
$$(x_i)_i \mapsto f_0(x_0) \oplus \cdots \oplus f_{n-1}(x_{n-1})$$

$$\text{Pointwise}_{n\times n,f_i} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$$
$$\Big((x_i)_i, (y_i)_i\Big) \mapsto f_0(x_0, y_0) \oplus \cdots \oplus f_{n-1}(x_{n-1}, y_{n-1})$$

$$\text{Reduction}_{n,f_i} : \mathbb{R}^n \to \mathbb{R}$$
$$(x_i)_i \mapsto f_{n-1}(x_{n-1}, f_{n-2}(x_{n-2}, f_{n-3}(\ldots f_0(x_0, \text{id}())\ldots)$$

$$\text{Induction}_{n,f_i} : \mathbb{R} \to \mathbb{R}^n$$
$$x \mapsto (\text{id}, f_1(x, \text{id}), f_2(x, f_1(x, \text{id})), \ldots)^\top$$

$$\text{Atomic}_{f(.)} : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto f(x)$$

$$\text{Atomic}_{f(.,.)} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$
$$(x, y) \mapsto f(x, y)$$

$$\text{Scale}_{n,(a,b)\mapsto a\diamond b} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$$
$$\Big(\alpha, (x_i)_{i=0,\ldots,n-1}\Big) \mapsto (\alpha \diamond x_i)_{i=0,\ldots,n-1}$$

Formal Code Synthesis

- **HCOL Breakdown Rules**

$$\text{SafeDist}_{V,A,b,\varepsilon}(.,.,.) \rightarrow \Big(P[x,(a_0,a_1,a_2)](.) < d_\infty^2(.,.)\Big)(.,.,.)$$

$$\text{with} \quad a_0 = \tfrac{1}{2b},\ a_1 = \tfrac{V}{b} + \varepsilon\Big(\tfrac{A}{b}+1\Big),\ a_2 = \Big(\tfrac{A}{b}+1\Big)\Big(\tfrac{A}{2}\varepsilon^2 + \varepsilon V\Big)$$

$$d_\infty^n(.,.) \rightarrow \|.\|_\infty^n \circ (-)_n$$

$$(\diamond)_n \rightarrow \text{Pointwise}_{n\times n,(a,b)\mapsto a\diamond b}$$

$$\|.\|_\infty^n \rightarrow \text{Reduction}_{n,(a,b)\mapsto \max(|a|,|b|)}$$

$$< .,. >_n \rightarrow \text{Reduction}_{n,(a,b)\mapsto a+b} \circ \text{Pointwise}_{n\times n,(a,b)\mapsto ab}$$

$$P[x,(a_0,\ldots,a_n)] \rightarrow < (a_0,\ldots,a_n),. > \circ (x^i)_n$$

$$(x^i)_n \rightarrow \text{Induction}_{n,(a,b)\mapsto ab,1}$$

- **Fully Expanded HCOL Expression**

$$\text{SafeDist}_{V,A,b,\varepsilon} \rightarrow \text{Atomic}_{(x,y)\mapsto x<y} \circ \Big(\Big(\text{Reduction}_{3,(x,y)\mapsto x+y} \circ \text{Pointwise}_{3,x\mapsto a_i x} \circ \text{Induction}_{3,(a,b)\mapsto ab,1}\Big) \times \Big(\text{Reduction}_{2,(x,y)\mapsto \max(|x|,|y|)} \circ \text{Pointwise}_{2\times 2,(x,y)\mapsto x-y}\Big)\Big)$$

HCOL and Σ–HCOL

- **HCOL problem specification:** *declarative, point free*

$$\text{ForAny}_{i=0}^{k-1}\left(\text{InsidePoly}_{m,n}(\mathbf{A}_i, \mathbf{b}_i, .)\right)$$

- **HCOL expression:** *data flow graph*

$$\text{Reduction}_{n,(a,b)\mapsto a\vee b} \circ \left[\begin{matrix}\cdot\\ \cdot\end{matrix}\right]_{\ell=0}^{k-1} \Big(\text{Reduction}_{n,(a,b)\mapsto a\wedge b} \circ \text{Pointwise}_{n,x_i\mapsto x_i\leq 0} \circ \text{Pointwise}_{n,x_i\mapsto x_i-b_i} \circ \left[\begin{matrix}\cdot\\ \cdot\end{matrix}\right]_{i=0}^{m-1} \left(\text{Reduction}_{n,(a,b)\mapsto a+b} \circ \text{Pointwise}_{n,x_j\mapsto a_{i,j}x_j}\right)\Big)$$

- **Σ-HCOL expression:** *optimized loop structure*

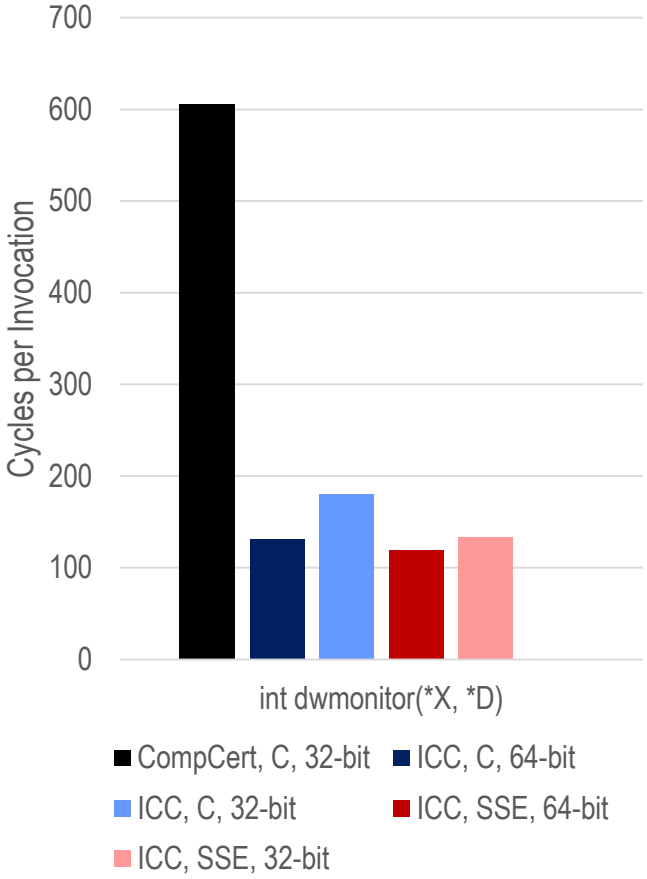
$$\sum_{\ell=0}^{k-1}\mathbb{V}\left(\sum_{i=0}^{m-1}\mathbb{\wedge}\left(\text{Atomic}_{x\mapsto(x-b_i)\leq 0}\sum_{j=0}^{n-1}\mathbb{+}\left(\text{Atomic}_{x\mapsto a_{i,j}x}\circ(\mathrm{e}_j^i)\top\right)\right)\right)$$

Final Synthesized Interval Arithmetic Code

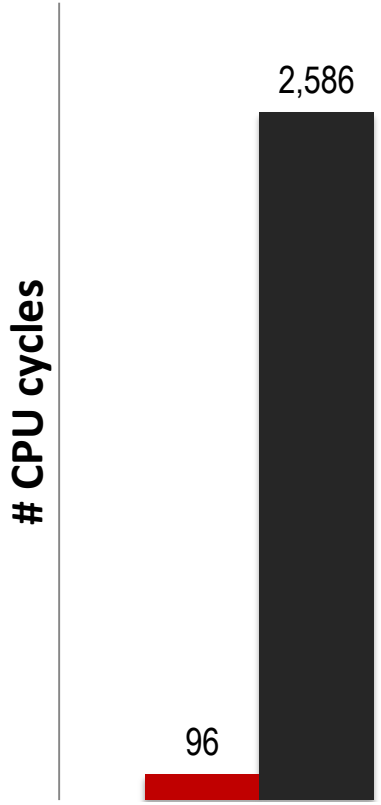
```
int dwmonitor(float  *X, double  *D) {
    __m128d u1, u2, u3, u4, u5, u6, u7, u8 , x1, x10, x13, x14, x17, x18, x19, x2, x3, x4, x6, x7, x8, x9;
    int w1;
    {
        unsigned _xm = _mm_getcsr();
        _mm_setcsr(_xm & 0xffff0000 | 0x0000dfc0);
        u5 = _mm_set1_pd(0.0);
        u2 = _mm_cvtps_pd(_mm_addsub_ps(_mm_set1_ps(FLT_MIN), _mm_set1_ps(X[0])));
        u1 = _mm_set_pd(1.0, (-1.0));
        for(int i5 = 0; i5 <= 2; i5++) {
            x6 = _mm_addsub_pd(_mm_set1_pd((DBL_MIN + DBL_MIN)), _mm_loaddup_pd(&(D[i5])));
            x1 = _mm_addsub_pd(_mm_set1_pd(0.0), u1);
            x2 = _mm_mul_pd(x1, x6);
            x3 = _mm_mul_pd(_mm_shuffle_pd(x1, x1, _MM_SHUFFLE2(0, 1)), x6);
            x4 = _mm_sub_pd(_mm_set1_pd(0.0), _mm_min_pd(x3, x2));
            u3 = _mm_add_pd(_mm_max_pd(_mm_shuffle_pd(x4, x4, _MM_SHUFFLE2(0, 1)), _mm_max_pd(x3, x2)), _mm_set1_
            u5 = _mm_add_pd(u5, u3);
            x7 = _mm_addsub_pd(_mm_set1_pd(0.0), u1);
            x8 = _mm_mul_pd(x7, u2);
            x9 = _mm_mul_pd(_mm_shuffle_pd(x7, x7, _MM_SHUFFLE2(0, 1)), u2);
            x10 = _mm_sub_pd(_mm_set1_pd(0.0), _mm_min_pd(x9, x8));
            u1 = _mm_add_pd(_mm_max_pd(_mm_shuffle_pd(x10, x10, _MM_SHUFFLE2(0, 1)), _mm_max_pd(x9, x8)), _mm_set
        }
        u6 = _mm_set1_pd(0.0);
        for(int i3 = 0; i3 <= 1; i3++) {
            u8 = _mm_cvtps_pd(_mm_addsub_ps(_mm_set1_ps(FLT_MIN), _mm_set1_ps(X[(i3 + 1)])));
            u7 = _mm_cvtps_pd(_mm_addsub_ps(_mm_set1_ps(FLT_MIN), _mm_set1_ps(X[(3 + i3)])));
            x14 = _mm_add_pd(u8, _mm_shuffle_pd(u7, u7, _MM_SHUFFLE2(0, 1)));
            x13 = _mm_shuffle_pd(x14, x14, _MM_SHUFFLE2(0, 1));
            u4 = _mm_shuffle_pd(_mm_min_pd(x14, x13), _mm_max_pd(x14, x13), _MM_SHUFFLE2(1, 0));
            u6 = _mm_shuffle_pd(_mm_min_pd(u6, u4), _mm_max_pd(u6, u4), _MM_SHUFFLE2(1, 0));
        }
        x17 = _mm_addsub_pd(_mm_set1_pd(0.0), u6);
        x18 = _mm_addsub_pd(_mm_set1_pd(0.0), u5);
        x19 = _mm_cmpge_pd(x17, _mm_shuffle_pd(x18, x18, _MM_SHUFFLE2(0, 1)));
        w1 = (_mm_testc_si128(_mm_castpd_si128(x19), _mm_set_epi32(0xffffffff, 0xffffffff, 0xffffffff, 0xffffffff
             (_mm_testnzc_si128(_mm_castpd_si128(x19), _mm_set_epi32(0xffffffff, 0xffffffff, 0xffffffff, 0xffffff
```

Spiral Interval Arithmetic Code Quality

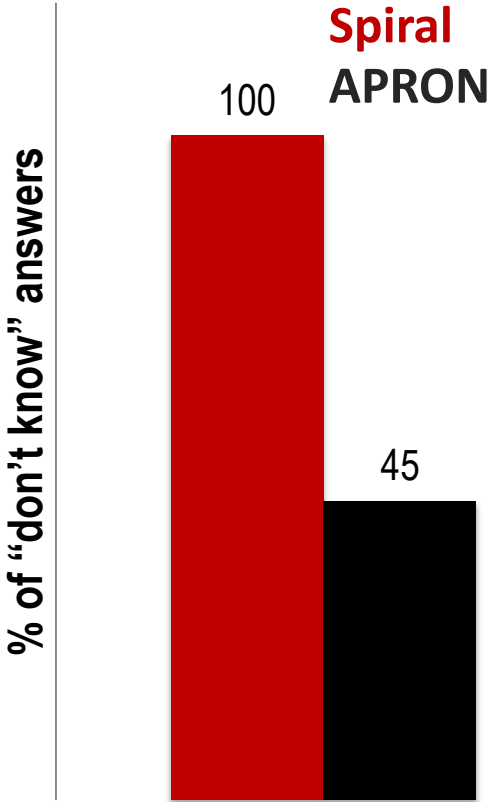
Intel C vs. CompCert



Performance



Precision at boundary



$a < b$? for
$a - b \approx 10^{-15}$

SandyBridge CPU, Intel C Compiler, CompCert, APRON Interval Arithmetic Library

High Assurance Spiral In A Nutshell

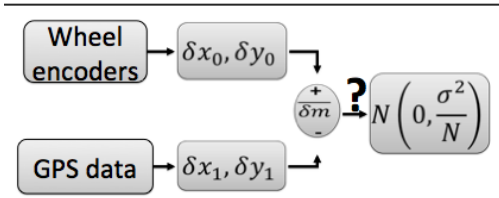
Problem and main idea

Co-synthesize high-quality code and proof for sensor-fusion based self-consistency algorithms

Deployed in Landshark robot

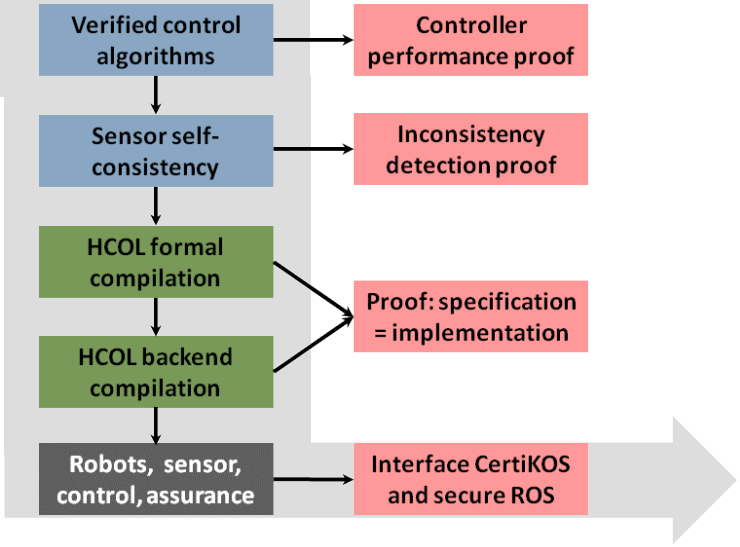
Approach

$$\mathbb{R} \times \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{Z}_2$$
$$(v_r, p_r, p_o) \mapsto \left(p(v_r) < d_\infty(p_r, p_o)\right)$$



$$A_i \mathcal{F}(\vec{x}) - b_i \preceq \vec{0}$$

$$\mathbf{x}^{t+h} \approx \left[\mathrm{I}_2 \,|h\, \mathrm{I}_2\right](\mathbf{x}^t \oplus \mathbf{v}^{t+h})$$




```
int dwmonitor(float  *X, double  *D)
    __m128d u1, u2, u3, u4, u5, u6,
    unsigned _xm = _mm_getcsr();
    _mm_setcsr(_xm & 0xffff0000 | 0x
    u5 = _mm_set1_pd(0.0);
    u2 = _mm_cvtps_pd(_mm_addsub_ps(
    _mm_set1_ps(FLT_MIN), _mm_set1_p
    u1 = _mm_set_pd(1.0, (-1.0));
    for(int i5 = 0; i5 <= 2; i5++) {
        x6 = _mm_addsub_pd(_mm_set1_
            +DBL_MIN)), _mm_loaddup_
        x1 = _mm_addsub_pd(_mm_set1_
        x2 = _mm_mul_pd(x1, x6);
    ...
        }
    __asm nop;
    if (_mm_getcsr() & 0x0d) {
        _mm_setcsr(_xm);
```

More Information:

www.spiral.net

www.spiralgen.com