Efficient Decision Procedure for Non-linear Arithmetic Constraints using CORDIC

Malay K. Ganai

Franjo Ivancic

System Analysis and Verification Group

NEC Labs America, Princeton, NJ

FMCAD 2009

Austin, TX

16-Nov-2009

Outline

- Introduction
- Related Work
- Background
  - CORDIC algorithms
- Our Approach: CORD
  - Encoding
  - DPLL-style Interval Search Engine (DISE)
- Experiments
- Conclusions

Introduction

- Non-linear problems arise in verification of hybrid discrete-continuous
  - Boolean combination of linear and non-linear real operations
  - Such problems are in general un-decidable
  - In practice: finite precision and interval bounds are user-provided (soundness and completeness)
- Operation Research
  - Use of floating-point library
  - Speed is traded off with accuracy (acceptable)
- Verification
  - Accuracy can not be traded off (undesirable)
  - Use of floating-point library or precise arithmetic

Related Work

- ❑ Absolver (Bauer et al. DATE 2007)
  - ➢ Boolean solver combined with off-the-shelf theory solver for linear and (imprecise) non-linear (IPOPT)
  - ➢ Result is neither sound nor complete

- ❑ LBR (Ganai HVC 2009)
  - ➢ Lazy bounding refinement using SMT-(LIA) solver
  - ➢ Restricted to bounded integer

Related Work

- iSAT (Franzle *et al.* JSBMC 2007)
  - Use interval constraint propagation
  - Use of floating-point library
  - Anomalous results observed (unacceptable)

Incompleteness?

$\varphi_1 := (x+y < a) \wedge (x-y < b) \wedge (2 \bullet x > a+b) \wedge (a = 1) \wedge (b = 0.1)$

φ_1 is UNSAT but iSAT(φ_1) declares SAT (spurious)

Soundness?

$\varphi_2 := (x \leq 10^9) \wedge (x+p > 10^9) \wedge (p = 10^{-8})$

φ_2 is SAT but iSAT(φ_2) declares UNSAT (unsound)

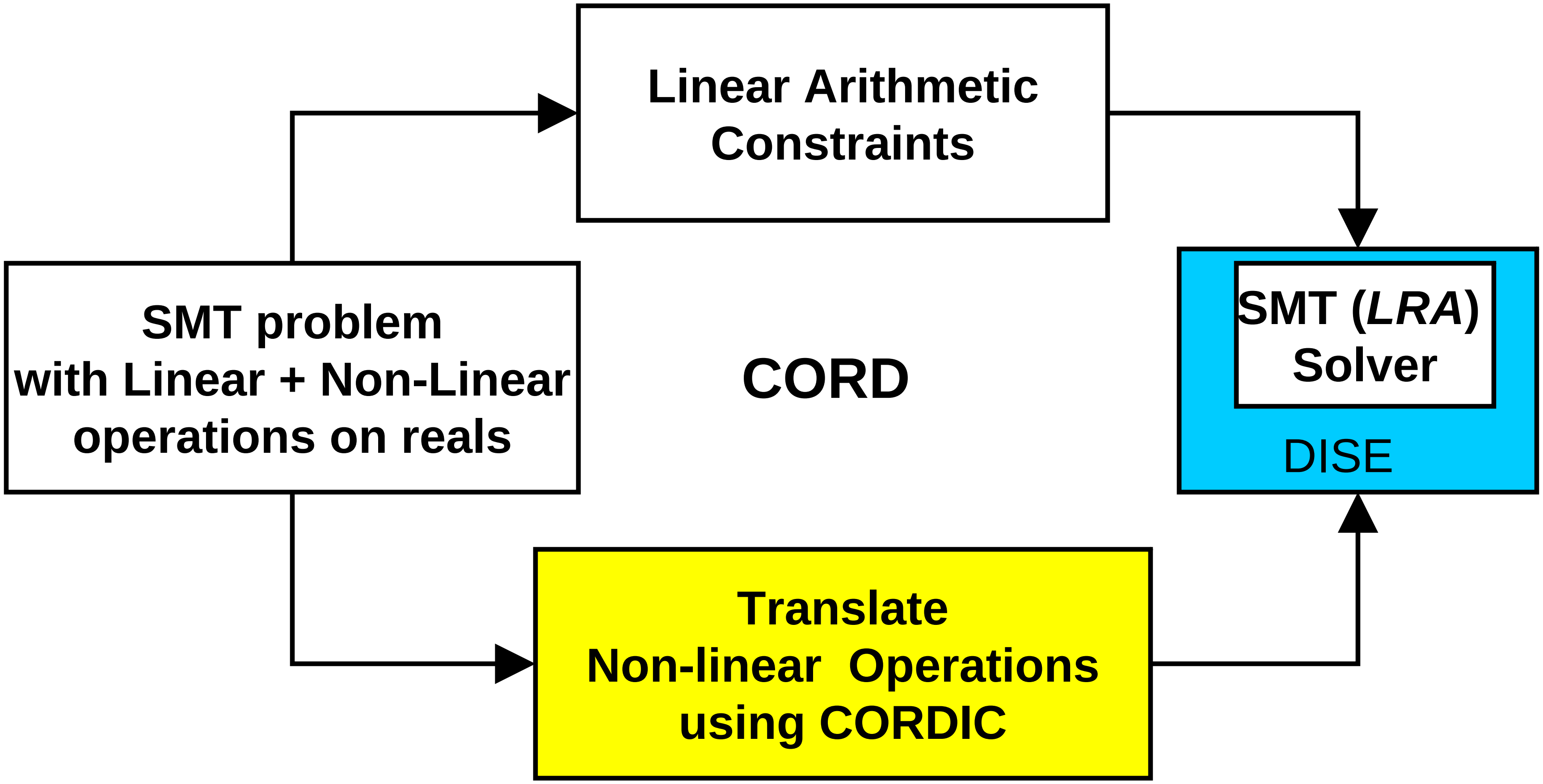
Motivation

```
 …
 x := y • y + 3 ° z; // x,y,z are real terms
 if (x > 10.03) {
     do_something1;
 } else {
     do_something2;
 }
…
```

Realization of real arithmetic is inherently inaccurate !

Unstable: For some x, $|x - 10.03| \leq \Delta$?

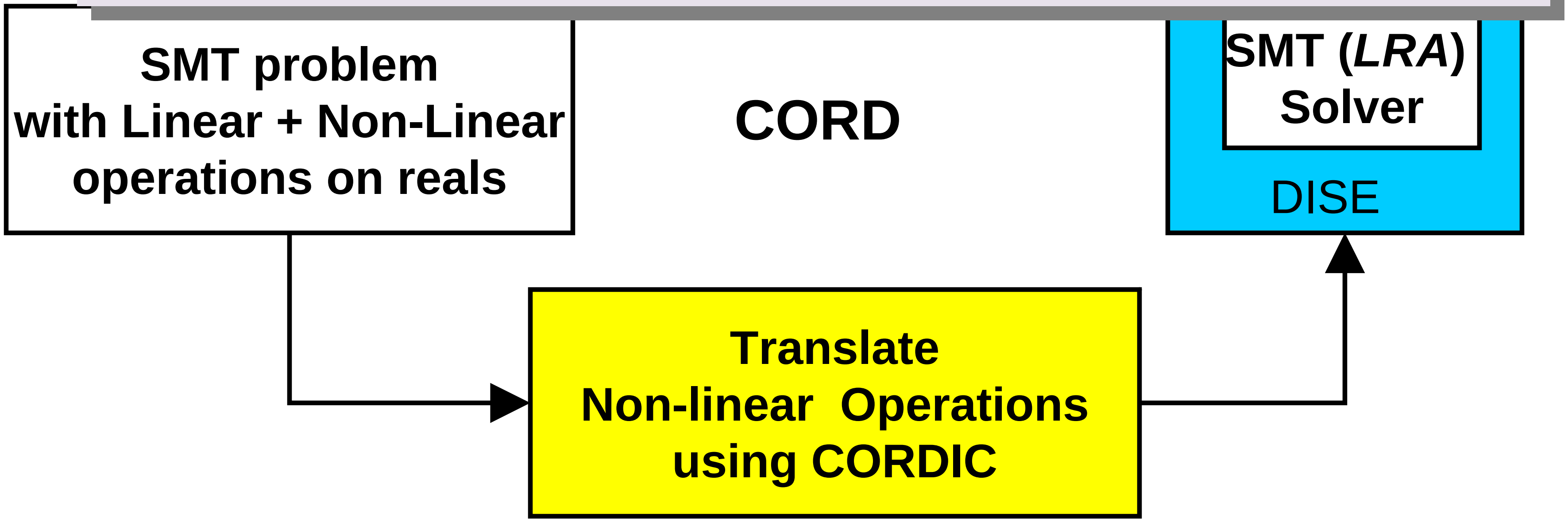
Our Decision Procedure: CORD


Linear Arithmetic Constraints
SMT problem with Linear + Non-Linear operations on reals
CORD
SMT (*LRA*) Solver
DISE
Translate Non-linear Operations using CORDIC

Our Decision Procedure: CORD

SOUND and COMPLETE

- ❑ Sound abstraction to account CORDIC induced inaccuracies safely
- ❑ Refinement through DISE



SMT problem with Linear + Non-Linear operations on reals

CORD

SMT (*LRA*) Solver

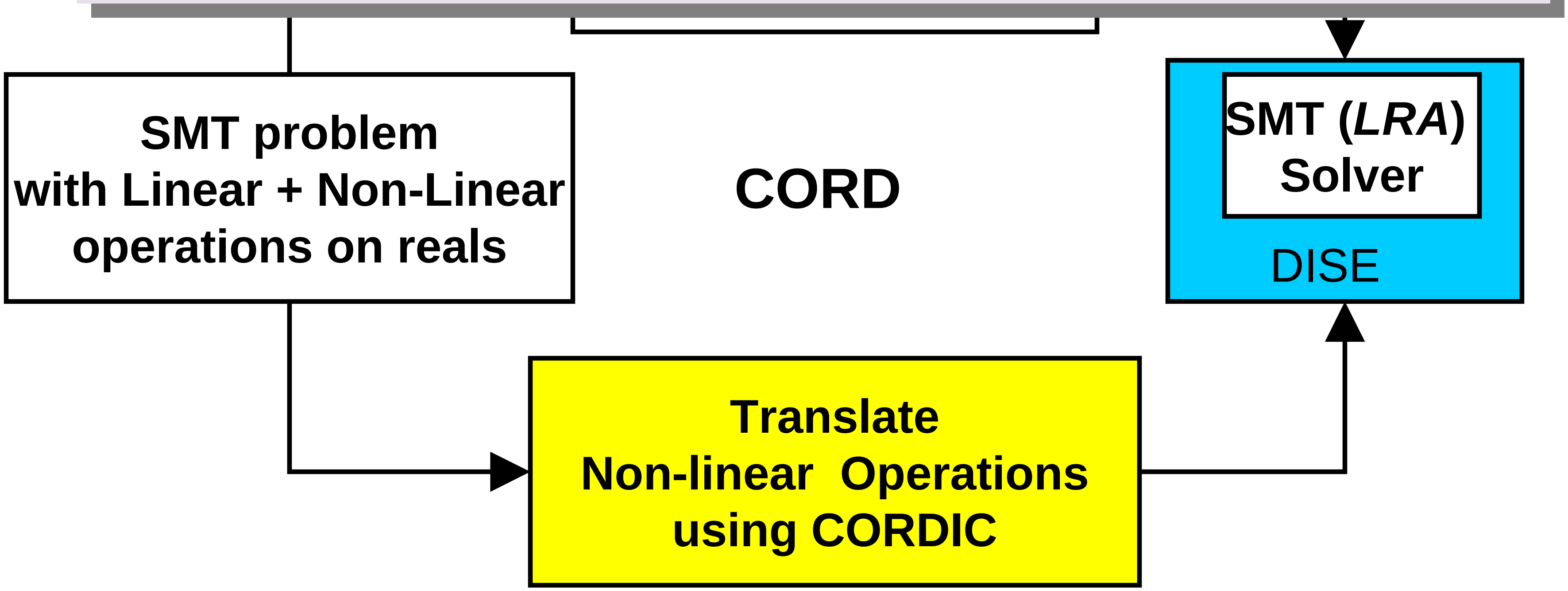
DISE

Translate Non-linear Operations using CORDIC

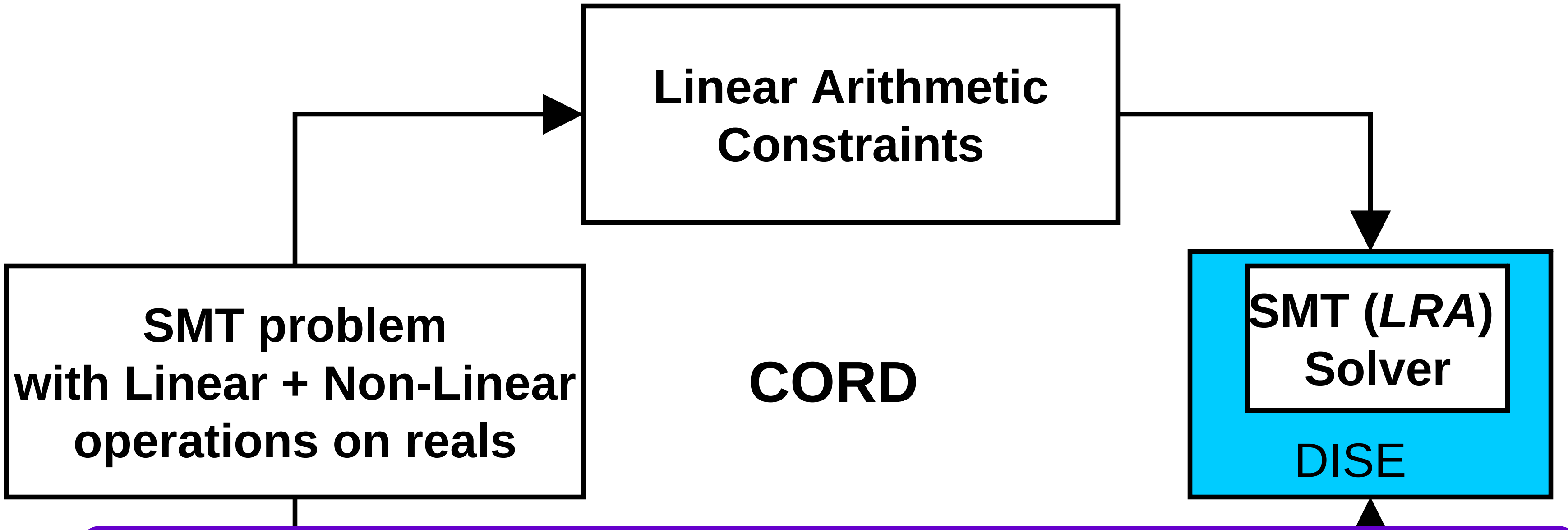
Our Decision Procedure: CORD

DPLL-style Interval Search Engine (DISE)

- ❑ Theory suggestions (≡ refinement “hints”)
- ❑ Theory Learning (≡ refinement steps)


SMT problem with Linear + Non-Linear operations on reals
CORD
SMT (*LRA*) Solver
DISE
Translate Non-linear Operations using CORDIC

Our Decision Procedure: CORD


Linear Arithmetic Constraints
SMT problem with Linear + Non-Linear operations on reals
CORD
SMT (*LRA*) Solver
DISE

Use of off-the-shelf SMT(LRA) solvers

- Precise (rational) arithmetic
- Leverage on the advancements of solvers

Outline

- Introduction
- Related Work
- Background
  - CORDIC algorithms
- Our Approach: CORD
  - Encoding
  - DPLL-style Interval Search Engine (DISE)
- Experiments
- Conclusions

What is CORDIC?

- **C**oordinate **R**otation **D**igital **C**omputer
  - (J. Volder in 1950)

- Used in Calculators, DSP fixed point operations

- For a given precision, it can compute wide range of elementary operations
  - multiplications, division, sin, cos, tan, square roots, log, exp

- Uses shift and add operators in a finite recursive formulations

CORDIC (in nutshell)

Algorithm (finitely recursive for 0≤ k ≤ n)

$$X[k+1] \leftarrow x[k] - c \circ \delta[c,k] \circ 2^{-\tau[c,k]} \circ y[k]$$

$$Y[k+1] \leftarrow y[k] + \delta[c,k] \circ 2^{-\tau[c,k]} \circ x[k]$$

$$Z[k+1] \leftarrow z[k] - \delta[c,k] \circ \alpha[c,k]$$

c	**τ[k]**	**α[k]**	**δ[k] = z[k] $\geq$ 0 ? 1: -1**	**δ[k] = y(k) $\geq$ 0 ? -1: 1**
0	k	2^{-k}	$y[n+1] \approx x[0] \bullet z[0]$ $(y[0] = 0)$	$z[n+1] \approx y[0] / x[0]$ $(z[0] = 0)$
1	k	$\tan^{-1} 2^{-k}$	$x[n+1] \approx \cos(z[0])$ $y[n+1] \approx \sin(z[0])$ $(x[0] = K, y[0] = 0)$	$z[n+1] \approx \tan^{-1}(y[0]/x[0])$ $x[n+1] \approx ((x[0])^2 + (y[0])^2)^{½} / K$ $(z[0] = 0)$
-1 k>0	k / k-1 (dep. on n)	$\tanh^{-1} 2^{-k}$	$x[n+1] \approx \cosh(z[1])$; $y[n+1] \approx \sinh(z[1])$ $x[n+1] + y[n+1] \approx e^{z[1]}$ $(x[1]=K', y[1]=0)$	$z[n+1] \approx \tanh^{-1}(y[1]/x[1])$ $x[n+1] \approx (x[1]^2 - y[1]^2)^{½}/K'$ $(x[1]>y[1], z[1] = 0)$

Outline

- Introduction
- Related Work
- Background
  - CORDIC algorithms
- Our Approach: CORD
  - Encoding
  - DPLL-style Interval Search Engine (DISE)
- Experiments
- Conclusions

Multiplication: $p = s \bullet t$

$\mathrm{CordMult}(s,t) = y[n] \approx p$

$$x[k+1] \leftarrow x[k] - 0 \circ \delta[0,k] \circ 2^{-k} \circ y[k]$$

$$y[k+1] \leftarrow y[k] + \delta[0,k] \circ 2^{-k} \circ x[k]$$

$$z[k+1] \leftarrow z[k] - \delta[0,k] \circ 2^{-k}$$

δ is $\mathrm{ITE}(z[k] \geq 0,1,-1)$, $z[0]=t$, $x[0]=s$

Errors: Absolute and Relative

$$\mathrm{Err}_{abs} = |y[n] - p|$$

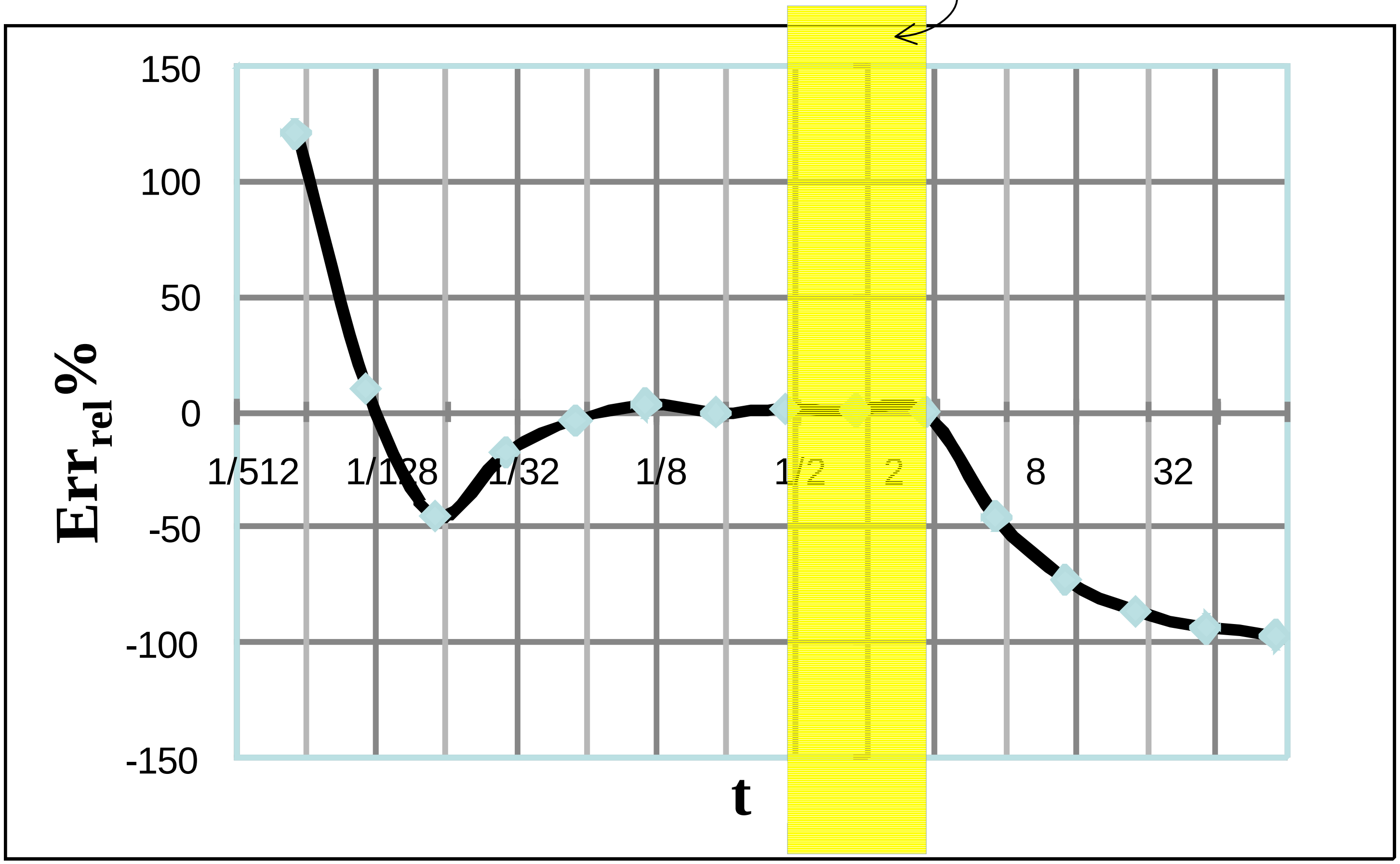
$$\mathrm{Err}_{rel} = \mathrm{Err}_{abs} / |p|$$

Quantization Error

$s = 0.567,\ t = 0.00355 \circ 2^r\ (0 \leq r < 10)$

$p = s \cdot t,\ p' =$ CordMult(s, t)

Err_{rel} is < 1% with n=8 in this range [1,2]



Error Bounds

Domain of Convergence (Wu et al TSP’92)

For $t \in [-2,2]$, errors are **bounded**

$$Err_{abs} = |\, z[n] \,| \bullet |\, x[0] \,| \leq 2^{-(n-1)} \circ |s|$$

$$Err_{rel} = |\, z[n] \,| \,/\, |\, z[0] \,| \leq 2^{-(n-1)} \,/\, |t|$$

- ➢ Rounding errors are not considered.
- ➢ We use precise linear arithmetic.

s=10, t=1.575, p=15.75, y[8]=15.7303135

$Err_{abs} = 0.04685 \leq 2^{-7} \circ 10 = 0.078$

$Err_{rel} = 0.00297 \leq 2^{-7} / 1.575 = 0.0049$

Larger Domain

Normalization (for | t | ≥ 2)

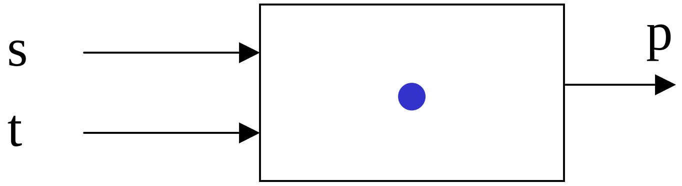
Let $t = Mt \circ 2^{Et}$ such that $|Mt| \in [1,2]$.

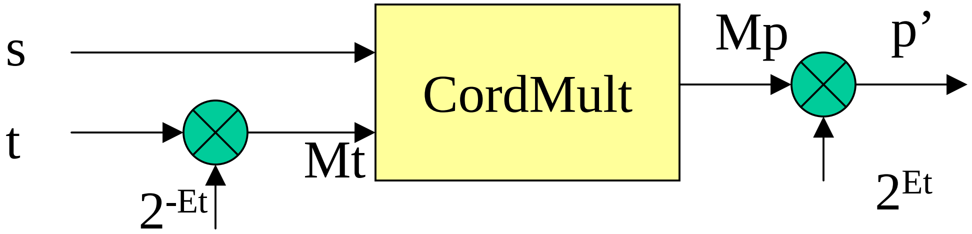
$p = s \bullet t$ can be cordic-translated as

$$Mp = \mathrm{CordMult}(s,Mt)$$

$$p \approx Mp \circ 2^{Et}$$

Bound on Err_{rel} depends only on n.





Error Tolerance (IEEE754-2008)

Error Tolerance δ

For given $\delta = 2^{-m}$, and we normalize t

$t = Mt \circ 2^{Et}$ with $|Mt| \in [1,2]$ and

$Et \geq -(m-n)$

$Err_{rel} = 2^{-(n-1)}$ for $t \geq 2^{-(m-n)}$

$Err_{abs} = 2^{-(m-1)} \circ | s |$ for $t \leq 2^{-(m-n)}$

If m >>n, clearly we reduce the size of cordic-iterative structure

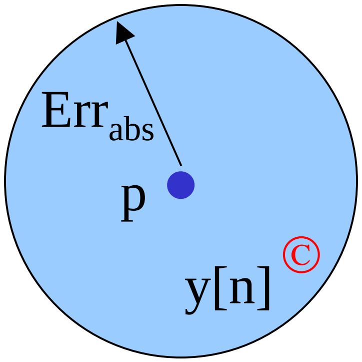
Over-approximation

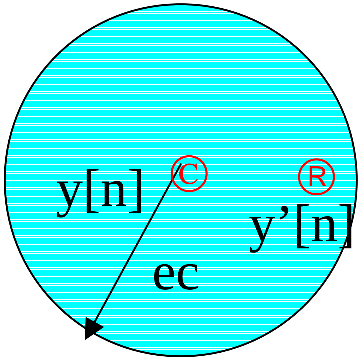
Error Correction

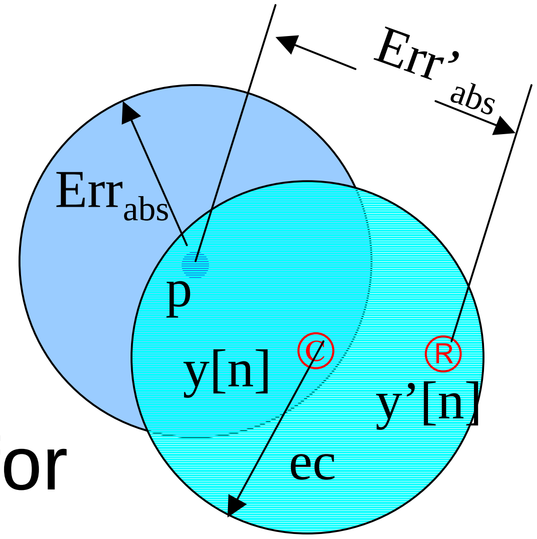
$$y'[n] = y[n] + ec, \text{ where } |ec| \le |s| \circ 2^{-(n-1)}$$

$$Err'_{abs} \le Err_{abs} + |s| \circ 2^{-(n-1)} \le |s| \circ 2^{-(n-2)}$$

$$Err'_{rel} = Err'_{abs} / |p| \le 2^{-(n-2)} / |t|$$







- ➢One more extra iteration step for same precision requirement.

Notation

Given $\varphi := \mathbf{B} \cup \mathbf{R}_{L} \cup \mathbf{R}_{NL} \cup \mathbf{R}_{ITE}$

B := {b | b is a Boolean expr on Boolean terms or predicate (=,<,>,≤,≥) on real terms}

R_L := {r| r is a linear term expr using °,+,- on real terms}

R_{NL} := {r| r is a non-linear term expr using •,/ on real terms}

R_{ITE} := {r| r= b ? r-term1 : r-term2}

Example (formula φ)

$(a <> (x \geq 3 \circ y+z))$ $//\mathbf{B},\ \mathbf{R}_{\mathbf{L}}$

$\wedge\ (s = \mathrm{ITE}(a,x,z))$ $//\mathbf{R}_{\mathbf{ITE}}$

$\wedge\ (t = \mathrm{ITE}(a,y,z)))$ $//\mathbf{R}_{\mathbf{ITE}}$

$\wedge\ (p = s \bullet t)$ $//\mathbf{R}_{\mathbf{NL}}$

$\wedge\ (p \geq (x+z))$ $//\mathbf{B},\ \mathbf{R}_{\mathbf{L}}$

CORDIC Translation

$\varphi \rightarrow \varphi'$

1) $B' := B$, $R_L' \leftarrow R_L$, $R'_{ITE} \leftarrow R_{ITE}$

2) For each $p = s \bullet t$ ($\in R_{NL}$)

   - add new real-vars: Mt, ec
   - $Mp = \text{CordMult}(s, Mt) + ec$
   - add constraints ($|ec| \leq s \circ 2^{-(n-1)}$),
     ($1 \leq |Mt| \leq 2$)
   - update B', R_L', R_{ITE}'

Example (translating…)

$(a <> (x \geq 3 \circ y+z))$

$\wedge\ (s = \mathrm{ITE}(a,x,z))$

$\wedge\ (t = \mathrm{ITE}(a,y,z))$

~~$\wedge\ (p = s \bullet t)$~~

$\wedge\ (p \geq (x+z))$

// new vars: Mt, ec

$\wedge\ (Mp = \mathrm{CordMult}(s,Mt))+ ec)$

$\wedge\ (|ec| \leq s \circ 2^{-(n-1)})$

$\wedge\ (1 \leq |Mt| \leq 2)$

Additional Constraints

Normalization(for each mult)

$$\mathrm{NC}\begin{cases}(t=Mt\circ 2^{Et})\wedge\ (p=Mp\circ 2^{Et}) & Et\ \geq -(m-n) \\ (t=Mt\circ 2^{-(m-n)})\ \wedge\ (p=Mp\circ 2^{-(m-n)}) & o.w.\end{cases}$$

Example (translating….)

(a <> (x ≥ 3 ° y+z))

∧ (s = ITE(a,x,z))

∧ (t = ITE(a,y,z))

~~∧ (p = s • t)~~

∧ (p ≥ (x+z))

// new real vars: Mt, ec

∧ (Mp = CordMult(s,Mt))+ ec)

∧ (|ec| ≤ s ° $2^{-(n-1)}$)

∧ (1 ≤ |Mt| ≤ 2)

// normalization const.

∧ (p = ITE(Et>-(m-n),Mp ° 2^{Et}, Mp ° $2^{-(m-n)}$))

∧ (t = ITE(Et>-(m-n),Mt ° 2^{Et}, Mt ° $2^{-(m-n)}$))

Additional Constraints

Interval Bounds (for each mult)

$$\mathrm{IB}\begin{cases} 2^{Et} \leq |t| \leq 2^{Et+1} & Et \geq -(m-n) \\ |t| \leq 2^{-(m-n)} & \text{otherwise} \end{cases}$$

Example (translating…done)

$(a <> (x \geq 3 \circ y+z))$

$\wedge\ (s = \text{ITE}(a,x,z))$

$\wedge\ (t = \text{ITE}(a,y,z))$

~~$\wedge\ (p = s \bullet t)$~~

$\wedge\ (p \geq (x+z))$

$\wedge\ (Mp = \text{CordMult}(s,Mt))+ ec)$ // new vars: Mt, ec

$\wedge\ (|ec| \leq s \circ 2^{-(n-1)})$

$\wedge\ (1 \leq |Mt| \leq 2)$

// normalization constraints (NC)

$\wedge\ (p = \text{ITE}(Et>-(m-n),Mp \circ 2^{Et}, Mp \circ 2^{-(m-n)}))$

$\wedge\ (t = \text{ITE}(Et>-(m-n),Mt \circ 2^{Et}, Mt \circ 2^{-(m-n)}))$

//interval bound constraints (IB)

$\wedge\ (tl \leq |t| \leq tu)$

$\wedge\ (tl = \text{ITE}(Et>-(m-n),2^{Et}, 0))$

$\wedge\ (tu = \text{ITE}(Et>-(m-n), 2^{Et+1}, 2^{-(m-n)}))$

Correctness of Encoding

Theorem 1 (soundness of encoding)

$$\varphi \Rightarrow_{SAT} \exists IB \; \varphi' \wedge \bigwedge_i (IB_i \wedge NC_i)$$

Numerical Accuracy

$$Err_{rel} \begin{cases} 2^{-(n-2)} & Et \geq -(m-n) \\ 2^{-(n-2)}/|Mt| & \text{otherwise} \end{cases}$$

$$Err_{abs} \begin{cases} 2^{-(n-2)+Et} \circ |s| & Et \geq -(m-n) \\ 2^{-(m-2)} \circ |s| & \text{otherwise} \end{cases}$$

Outline

- Introduction
- Related Work
- Background
  - CORDIC algorithms
- Our Approach: CORD
  - Encoding
  - DPLL-style Interval Search Engine (DISE)
- Experiments
- Conclusions

DISE: Overview

Problem

$$\exists \mathtt{IB}\ \underbrace{\mathtt{xIB}}_{\text{External bounds on on non-linear terms}} \wedge \varphi' \wedge \wedge_i(\underbrace{\mathtt{IB}_i \wedge \mathtt{NC}_i}_{\text{Search over } Et_i})$$

Our Solution

Solve the interval search problem incrementally in a DPLL-style, using theory suggestion and theory learning.

DISE: Search Strategy

$\psi = \psi_H \wedge \psi_S$

Hard (infinite-weights) Soft (finite-weights)

Theory Suggestion

If ψ is sat, so is ψ_H; but ψ_s may not be sat. Suggest new values/constraints.

Theory Learning

If ψ is unsat, so is ψ_H. Identify failed constraints, and add theory lemma.

DISE: Problem formulation

Hard and Soft Constraints

Let $C, U \subseteq \Omega = \{IB_1 \wedge NC_1, \ldots, IB_u \wedge NC_u\}$

$$\psi = \underset{\text{Learnt}}{\underset{\downarrow}{\Gamma}} \wedge \varphi' \wedge xIB \wedge \underset{\text{Committed}}{\underset{\downarrow}{(\wedge_{\omega \in C}\, \omega)}} \wedge \underset{\text{uncommitted}}{\underset{\downarrow}{(\wedge_{\nu \in U}\, \nu)}}$$

```
Initially, $C=\phi$, $U=\Omega$, stop=0
 while(!stop) {
     tmp_result          =  SMT(LRA)_Solve($\psi$);
    (result, stop, $\psi$)   =  Refine_Check($\psi$, tmp_result));
  }
 return result
```

DISE: Refine_Check

$$\psi = \Gamma \wedge \varphi' \wedge \mathtt{xIB} \wedge (\bigwedge_{\omega \in C} \omega) \wedge (\bigwedge_{\nu \in U} \nu)$$

- ❑ Case 1: ψ **is SAT and** $\boldsymbol{U}=\phi$ return SAT
- ❑ Case 2: ψ **is UNSAT and** $\boldsymbol{C}=\phi$ return UNSAT
- ❑ Case 3: ψ **is SAT**
  - ➢ If all ***U*** is SAT, return SAT
  - ➢ Otherwise, update interval ν s.t. ν is satisfied. Update U.
  - ➢ Pick $\nu \in$ ***U*** as next decision variable. Update C and U.
- ❑ Case 4: ψ **is UNSAT and** $\boldsymbol{C} \neq \phi$
  - ➢ Find sufficient set of infeasible constraints $\boldsymbol{IC} \subseteq \boldsymbol{C}$. Add lemma in Γ
  - ➢ Identify backtrack level, backtrack

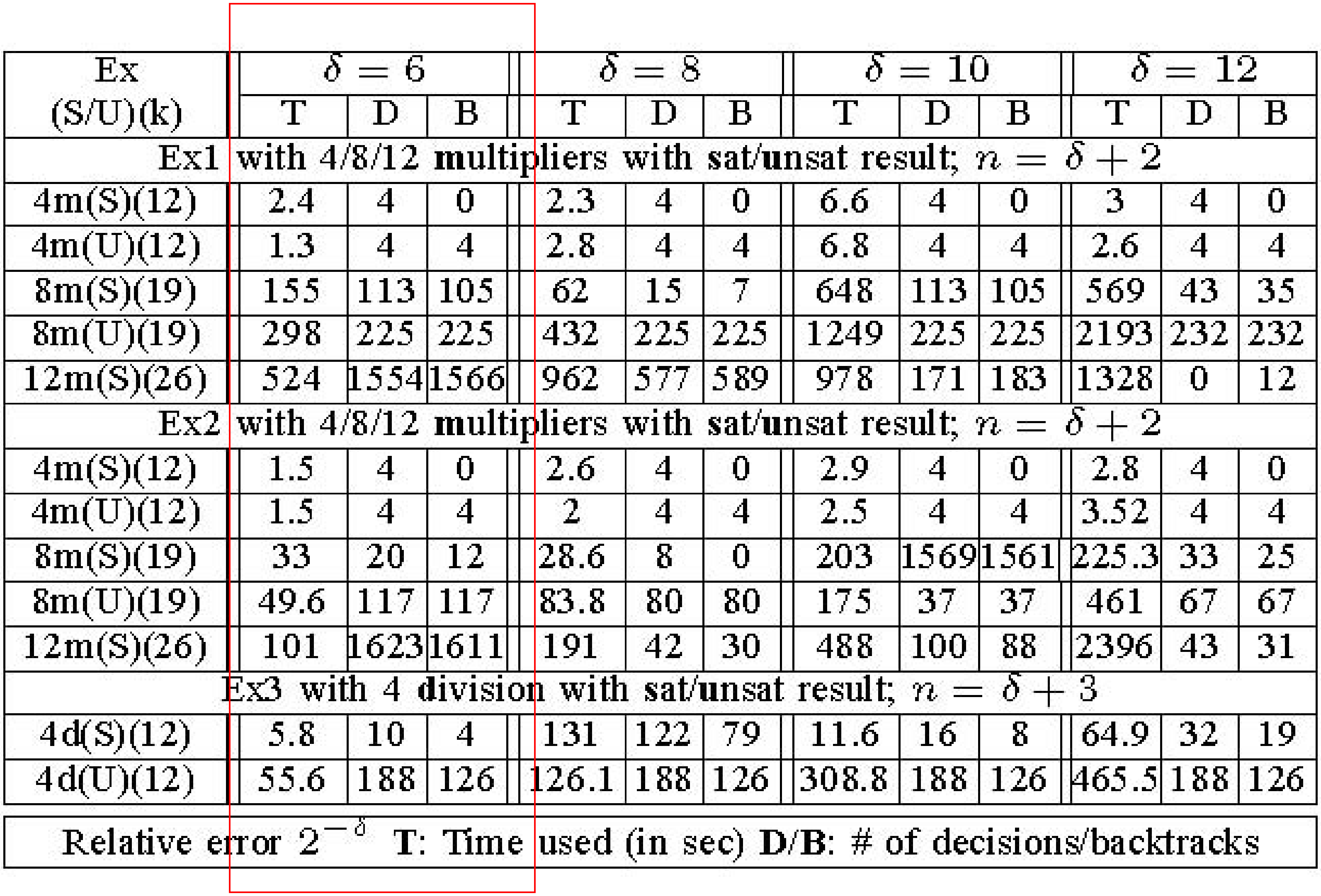
Correctness of CORD

Theorem 2

CORD always terminates correctly, either with an unsat result or a sat result with in the given precision specified.

Proof is available at
http://www.nec-labs.com/~malay/notes.htm

Experiments

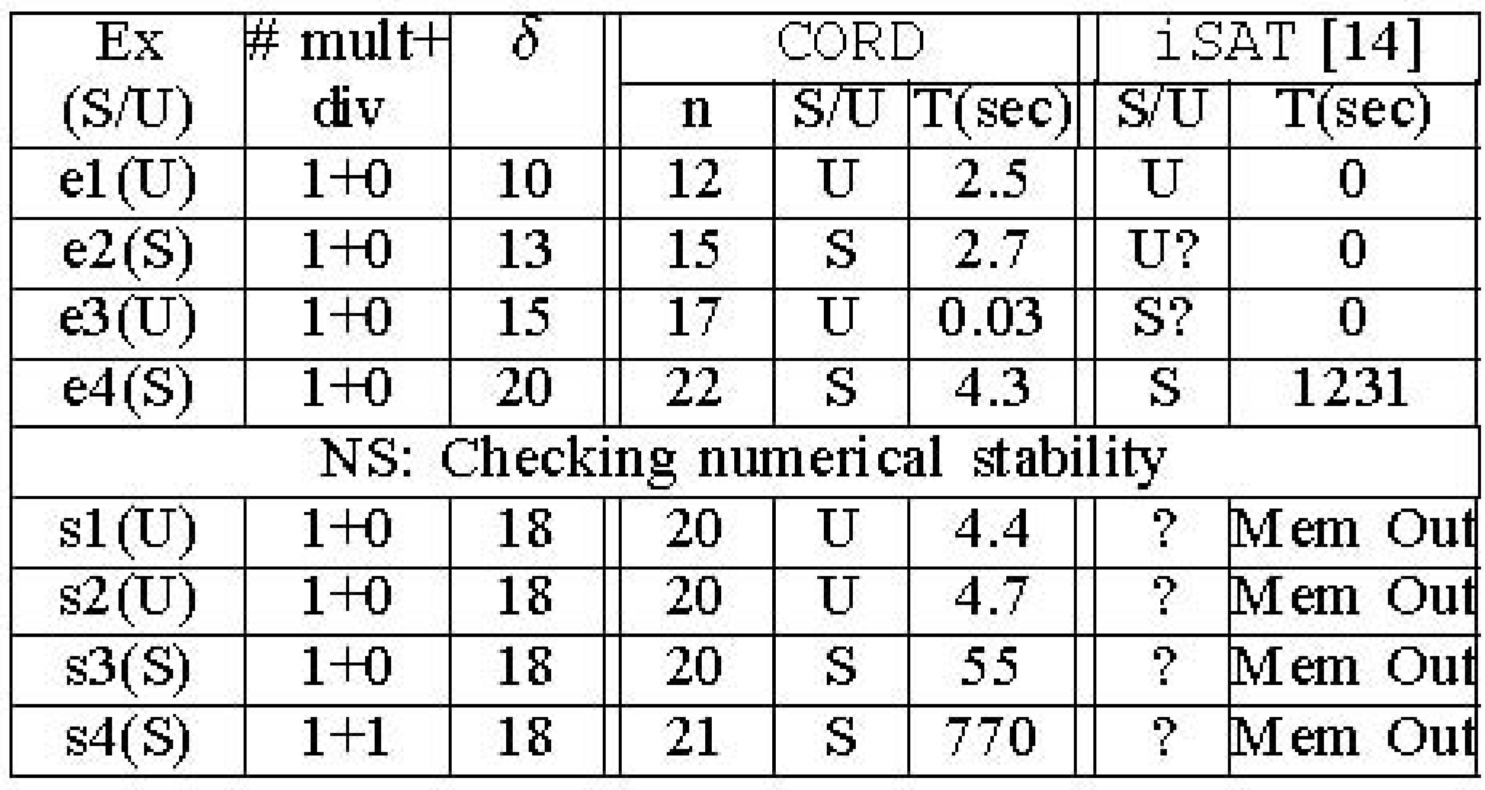


Ex (S/U)(k)	$\delta = 6$			$\delta = 8$			$\delta = 10$			$\delta = 12$		
	T	D	B	T	D	B	T	D	B	T	D	B
Ex1 with 4/8/12 **multipliers** with **sat/unsat result**; $n = \delta + 2$												
4m(S)(12)	2.4	4	0	2.3	4	0	6.6	4	0	3	4	0
4m(U)(12)	1.3	4	4	2.8	4	4	6.8	4	4	2.6	4	4
8m(S)(19)	155	113	105	62	15	7	648	113	105	569	43	35
8m(U)(19)	298	225	225	432	225	225	1249	225	225	2193	232	232
12m(S)(26)	524	1554	1566	962	577	589	978	171	183	1328	0	12
Ex2 with 4/8/12 **multipliers** with **sat/unsat result**; $n = \delta + 2$												
4m(S)(12)	1.5	4	0	2.6	4	0	2.9	4	0	2.8	4	0
4m(U)(12)	1.5	4	4	2	4	4	2.5	4	4	3.52	4	4
8m(S)(19)	33	20	12	28.6	8	0	203	1569	1561	225.3	33	25
8m(U)(19)	49.6	117	117	83.8	80	80	175	37	37	461	67	67
12m(S)(26)	101	1623	1611	191	42	30	488	100	88	2396	43	31
Ex3 with 4 **division** with **sat/unsat result**; $n = \delta + 3$												
4d(S)(12)	5.8	10	4	131	122	79	11.6	16	8	64.9	32	19
4d(U)(12)	55.6	188	126	126.1	188	126	308.8	188	126	465.5	188	126

Relative error $2^{-\delta}$ **T**: Time used (in sec) **D/B**: # of decisions/backtracks

Platform: intel 3.4Ghz 2Gb RAM running linux. Yices SMT solver

Experiment (CORD vs iSAT)



Ex (S/U)	# mult+div	δ	CORD			iSAT [14]	
			n	S/U	T(sec)	S/U	T(sec)
e1(U)	1+0	10	12	U	2.5	U	0
e2(S)	1+0	13	15	S	2.7	U?	0
e3(U)	1+0	15	17	U	0.03	S?	0
e4(S)	1+0	20	22	S	4.3	S	1231
NS: Checking numerical stability							
s1(U)	1+0	18	20	U	4.4	?	Mem Out
s2(U)	1+0	18	20	U	4.7	?	Mem Out
s3(S)	1+0	18	20	S	55	?	Mem Out
s4(S)	1+1	18	21	S	770	?	Mem Out

Examples at: http://www.nec-labs.com/research/system/
/systems_SAV-website/benchmarks.php

Conclusion/Future work

- ❑ Discussed an efficient encoding of non-linear arithmetic using CORDIC
- ❑ Discussed a sound and complete decision procedure based on DPLL-style interval search engine, with guiding mechanism
- ❑ Our formulation uses off-the-shelf SMT solvers for LRA, and therefore, can leverage from their ongoing advancements.
- ❑ In future, we extend current approach to handle other elementary operations with improved DPLL-style reasoning.