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?**

\[ \varphi_1 := (x+y < a) \land (x-y < b) \land (2 \cdot x > a+b) \land (a = 1) \land (b = 0.1) \]

\( \varphi_1 \) is UNSAT but iSAT(\( \varphi_1 \)) declares SAT (spurious)

**Soundness?**

\[ \varphi_2 := (x \leq 10^9) \land (x + p > 10^9) \land (p = 10^{-8}) \]

\( \varphi_2 \) is SAT but iSAT(\( \varphi_2 \)) declares UNSAT (unsound)
Motivation

...  
\[ x := y \cdot y + 3 \circ z; \quad // \quad x, y, z \text{ are real terms} \]
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
- Translate Non-linear Operations using CORDIC
- SMT (LRA) Solver
- DISE
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

Translate Non-linear Operations using CORDIC

SMT (LRA) Solver

DISE

Ganai et al  DP for non-linear arithmetic using CORDIC  FMCAD 2009 Austin
Our Decision Procedure: CORD

- SMT problem with Linear + Non-Linear operations on reals
- CORD
- Translate Non-linear Operations using CORDIC
- DPLL-style Interval Search Engine (DISE)
  - Theory suggestions (≡ refinement “hints”)
  - Theory Learning (≡ refinement steps)

SMT (LRA) Solver
DISE
Our Decision Procedure: CORD

SMT problem with Linear + Non-Linear operations on reals

Linear Arithmetic Constraints

SMT (LRA) Solver

CORD

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?

- **Coordinate Rotation Digital Computer**
  - (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 $0 \leq k \leq n$)

\[
\begin{align*}
X[k+1] & \leftarrow x[k] - \delta[c,k] \circ 2^{-\tau[c,k]} \circ y[k] \\
Y[k+1] & \leftarrow y[k] + \delta[c,k] \circ 2^{-\tau[c,k]} \circ x[k] \\
Z[k+1] & \leftarrow z[k] - \delta[c,k] \circ \alpha[c,k]
\end{align*}
\]

<table>
<thead>
<tr>
<th>c</th>
<th>$\tau[k]$</th>
<th>$\alpha[k]$</th>
<th>$\delta[k] = z[k] \geq 0 \ ? \ 1: -1$</th>
<th>$\delta[k] = y(k) \geq 0 \ ? \ -1: 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>k</td>
<td>$2^{-k}$</td>
<td>$y[n+1] \approx x[0] \cdot z[0]$ ($y[0] = 0$)</td>
<td>$z[n+1] \approx y[0] / x[0]$ ($z[0] = 0$)</td>
</tr>
<tr>
<td>1</td>
<td>k</td>
<td>$\tan^{-1}2^{-k}$</td>
<td>$x[n+1] \approx \cos(z[0])$ $y[n+1] \approx \sin(z[0])$ ($x[0] = K$, $y[0] = 0$)</td>
<td>$z[n+1] \approx \tan^{-1}(y[0]/x[0])$ $x[n+1] \approx (x[0])^2 + (y[0])^2 \frac{1}{2} / K$ ($z[0] = 0$)</td>
</tr>
<tr>
<td>-1</td>
<td>k / k-1 (dep. on n)</td>
<td>$\tanh^{-1}2^{-k}$</td>
<td>$x[n+1] \approx \cosh(z[1])$; $y[n+1] \approx \sinh(z[1])$ $x[n+1] + y[n+1] \approx e^{z[1]}$ ($x[1] = K'$, $y[1] = 0$)</td>
<td>$z[n+1] \approx \tanh^{-1}(y[1]/x[1])$ $x[n+1] \approx (x[1])^2 - (y[1])^2 \frac{1}{2} / K'$ ($x[1] &gt; y[1]$, $z[1] = 0$)</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Related Work
- Background
  - CORDIC algorithms
- Our Approach: CORD
  - Encoding
  - DPLL-style Interval Search Engine (DISE)
- Experiments
- Conclusions
Multiplication: \( p = s \cdot t \)

\[
\text{CordMult}(s,t) = y[n] \approx p \\
x[k+1] \leftarrow x[k] - 0 \degree \delta[0,k] \cdot 2^{-k} \cdot y[k] \\
y[k+1] \leftarrow y[k] + \delta[0,k] \cdot 2^{-k} \cdot x[k] \\
z[k+1] \leftarrow z[k] - \delta[0,k] \cdot 2^{-k}
\]

\( \delta \) is ITE(\( z[k] \geq 0,1,-1 \)), \( z[0]=t, \ x[0]=s \)

Errors: Absolute and Relative

\[
\text{Err}_{\text{abs}} = |y[n] - p| \\
\text{Err}_{\text{rel}} = \text{Err}_{\text{abs}} / |p|
\]
Quantization Error

\[ s = 0.567, \quad t = 0.00355 \cdot 2^r \quad (0 \leq r < 10) \]

\[ p = s \cdot t, \quad p' = \text{CordMult}(s, t) \]

\[ \text{Err}_{rel} \text{ is } < 1\% \text{ with } n=8 \text{ in this range } [1,2] \]
Error Bounds

Domain of Convergence (Wu et al TSP’92)

For \( t \in [-2,2] \), errors are bounded

\[
\begin{align*}
\text{Err}_{\text{abs}} &= |z[n]| \cdot |x[0]| \leq 2^{-(n-1)} \cdot |s| \\
\text{Err}_{\text{rel}} &= |z[n]| / |z[0]| \leq 2^{-(n-1)} / |t|
\end{align*}
\]

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

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

\[
\begin{align*}
\text{Err}_{\text{abs}} &= 0.04685 \leq 2^{-7} \cdot 10 = 0.078 \\
\text{Err}_{\text{rel}} &= 0.00297 \leq 2^{-7} / 1.575 = 0.0049
\end{align*}
\]
Larger Domain

Normalization (for $|t| \geq 2$)

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

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

$M_p = \text{CordMult}(s,M)$

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

Bound on $\text{Err}_{rel}$ depends only on $n$. 

Ganai et al. DP for non-linear arithmetic using CORDIC FMCAD 2009 Austin
Error Tolerance (IEEE754-2008)

Error Tolerance $\delta$

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

$$t = Mt \circ 2^{Et} \text{ with } |Mt| \in [1,2] \text{ and } Et \geq -(m-n)$$

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

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

If $m >> n$, clearly we reduce the size of cORDIC-iterative structure
Over-approximation

Error Correction

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

\[ \text{Err'}_{\text{abs}} \leq \text{Err}_{\text{abs}} + |s| \circ 2^{-(n-1)} \leq |s| \circ 2^{-(n-2)} \]

\[ \text{Err'}_{\text{rel}} = \frac{\text{Err'}_{\text{abs}}}{|p|} \leq 2^{-(n-2)} / |t| \]

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

Given \( \varphi := B \cup R_L \cup R_{NL} \cup R_{ITE} \)

\[ B := \{ b \mid b \text{ is a Boolean expr on Boolean terms or predicate } (=,\,<,\,>,\,\leq,\,\geq) \text{ on real terms} \} \]
\[ R_L := \{ r \mid r \text{ is a linear term expr using } °,+,- \text{ on real terms} \} \]
\[ R_{NL} := \{ r \mid r \text{ is a non-linear term expr using } \cdot,/ \text{ on real terms} \} \]
\[ R_{ITE} := \{ r \mid r= b \ ? \ r\text{-term1} : r\text{-term2} \} \]
Example (formula $\varphi$)

\[(a <> (x \geq 3 \circ y+z)) \quad // \mathbf{B, R_L} \]
\[\land (s = \text{ITE}(a, x, z)) \quad // \mathbf{R_{ITE}} \]
\[\land (t = \text{ITE}(a, y, z)) \quad // \mathbf{R_{ITE}} \]
\[\land (p = s \cdot t) \quad // \mathbf{R_{NL}} \]
\[\land (p \geq (x+z)) \quad // \mathbf{B, R_L} \]
CORDIC Translation

 φ → φ′

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

2) For each $p = s \cdot t \ (\in R_{NL})$
   - add new real-vars: $M_t, ec$
   - $M_p = \text{CordMult}(s, M_t) + ec$
   - add constraints $(|ec| \leq s \cdot 2^{-(n-1)})$
     $\quad (1 \leq |M_t| \leq 2)$
   - update $B′, R_L′, R_{ITE}′$
Example (translating...)

(a <> (x ≥ 3 ° y+z))
∧ (s = ITE(a, x, z))
∧ (t = ITE(a, y, z))
∧ (p = s • t)
∧ (p ≥ (x+z))
∧ (Mp = CordMult(s, Mt))+ ec)
∧ (|ec| ≤ s ° 2^{-(n-1)})
∧ (1 ≤ |Mt| ≤ 2)

// new vars: Mt, ec
Additional Constraints

\[
\text{Normalization (for each mult)}
\]

\[
\text{NC}
\]

\[
\begin{cases} 
(t = M_t \cdot 2^{E_t}) \land (p = M_p \cdot 2^{E_t}) & \text{if } E_t \geq -(m-n) \\
(t = M_t \cdot 2^{-(m-n)}) \land (p = M_p \cdot 2^{-(m-n)}) & \text{o.w.}
\end{cases}
\]
Example (translating....)

\[(a \not<\rangle (x \geq 3 \, ^\circ \, y+z))\]
\[\land (s = \text{ITE}(a,x,z))\]
\[\land (t = \text{ITE}(a,y,z))\]
\[\land (p = s \cdot t)\]
\[\land (p \geq (x+z))\]

\[// \text{new real vars: } Mt, \text{ ec}\]
\[\land (Mp = \text{CordMult}(s,Mt)) + \text{ ec})\]
\[\land (|\text{ec}| \leq s \, ^\circ \, 2^{-(n-1)})\]
\[\land (1 \leq |Mt| \leq 2)\]

\[// \text{normalization const.}\]
\[\land (p = \text{ITE}(E_t>-(m-n),Mp \, 2^{E_t}, Mp \, 2^{-(m-n)}))\]
\[\land (t = \text{ITE}(E_t>-(m-n),Mt \, 2^{E_t}, Mt \, 2^{-(m-n)}))\]
Additional Constraints

Interval Bounds (for each mult)

\[
\begin{align*}
\text{IB} & \quad \begin{cases} 
2^{E_t} \leq |t| \leq 2^{E_t+1} & E_t \geq -(m-n) \\
|t| \leq 2^{-(m-n)} & \text{otherwise}
\end{cases}
\end{align*}
\]
Example (translating...done)

\[(a \leftrightarrow (x \geq 3 \circ y+z))\]
\[\land (s = \text{ITE}(a,x,z))\]
\[\land (t = \text{ITE}(a,y,z))\]
\[\land (p = s \cdot t)\]
\[\land (p \geq (x+z))\]
\[\land (Mp = \text{CordMult}(s,Mt))+ \text{ec})  //\text{new vars:} \text{Mt}, \text{ec}\]
\[\land (|\text{ec}| \leq s \circ 2^{-(n-1)})\]
\[\land (1 \leq |\text{Mt}| \leq 2)\]

// normalization constraints (NC)
\[\land (p = \text{ITE}(\text{Et}>-(m-n),Mp \circ 2^\text{Et}, Mp \circ 2^{-(m-n)}))\]
\[\land (t = \text{ITE}(\text{Et}>-(m-n),Mt \circ 2^\text{Et}, Mt \circ 2^{-(m-n)}))\]

//interval bound constraints (IB)
\[\land (tl \leq |t| \leq tu)\]
\[\land (tl = \text{ITE}(\text{Et}>-(m-n), 2^\text{Et}, 0))\]
\[\land (tu = \text{ITE}(\text{Et}>-(m-n), 2^{\text{Et}+1}, 2^{-(m-n)}))\]
Correctness of Encoding

**Theorem 1** (soundness of encoding)

\[ \varphi \Rightarrow_{\text{SAT}} \exists \text{IB} \; \varphi' \land \land_i (\text{IB}_i \land \text{NC}_i) \]

**Numerical Accuracy**

\[
\begin{align*}
\text{Err}_{\text{rel}} & = \begin{cases} 
2^{-(n-2)} & \text{if } E_t \geq -(m-n) \\
2^{-(n-2)}/|M_t| & \text{otherwise}
\end{cases} \\
\text{Err}_{\text{abs}} & = \begin{cases} 
2^{-(m-2)} \circ |s| & \text{if } E_t \geq -(m-n) \\
2^{-(n-2)} \circ |s| & \text{otherwise}
\end{cases}
\end{align*}
\]
Outline

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

**Problem**

$$\exists IB \times IB \land \varphi' \land \land_i (IB_i \land NC_i)$$

- External bounds on non-linear terms
- 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

\[ \psi = \psi_H \land \psi_S \]

Hard (infinite-weights) \hspace{2cm} Soft (finite-weights)

Theory Suggestion

If \( \psi \) is sat, so is \( \psi_H \); but \( \psi_S \) may not be sat. **Suggest** new values/constraints.

Theory Learning

If \( \psi \) is unsat, so is \( \psi_H \). **Identify** failed constraints, and **add** theory lemma.
DISE: Problem formulation

Let $C, U \subseteq \Omega = \{ IB_1 \land NC_1, ..., IB_u \land NC_u \}$

$$\psi = \Gamma \land \varphi' \land x_{IB} \land (\land_{\omega \in C} \omega) \land (\land_{\nu \in U} \nu)$$

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

\[ \psi = \Gamma \land \phi' \land \text{IB} \land (\land_{\omega \in C} \omega) \land (\land_{v \in U} v) \]

- **Case 1:** \( \psi \) is SAT and \( U = \phi \) return SAT
- **Case 2:** \( \psi \) is UNSAT and \( C = \phi \) return UNSAT
- **Case 3:** \( \psi \) is SAT
  - If all \( U \) is SAT, return SAT
  - Otherwise, update interval \( v \) s.t. \( v \) is satisfied. Update \( U \).
  - Pick \( v \in U \) as next decision variable. Update \( C \) and \( U \).
- **Case 4:** \( \psi \) is UNSAT and \( C \neq \phi \)
  - Find sufficient set of infeasible constraints \( IC \subseteq C \). Add lemma in \( \Gamma \)
  - 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

<table>
<thead>
<tr>
<th>Ex (S/U)(k)</th>
<th>$\delta = 6$</th>
<th>$\delta = 8$</th>
<th>$\delta = 10$</th>
<th>$\delta = 12$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
<td>D</td>
<td>B</td>
<td>T</td>
</tr>
<tr>
<td>Ex1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4m(S)(12)</td>
<td>2.4</td>
<td>4</td>
<td>0</td>
<td>2.3</td>
</tr>
<tr>
<td>4m(U)(12)</td>
<td>1.3</td>
<td>4</td>
<td>4</td>
<td>2.8</td>
</tr>
<tr>
<td>8m(S)(19)</td>
<td>155</td>
<td>113</td>
<td>105</td>
<td>62</td>
</tr>
<tr>
<td>8m(U)(19)</td>
<td>298</td>
<td>225</td>
<td>225</td>
<td>432</td>
</tr>
<tr>
<td>12m(S)(26)</td>
<td>524</td>
<td>1554</td>
<td>1566</td>
<td>962</td>
</tr>
<tr>
<td>Ex2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4m(S)(12)</td>
<td>1.5</td>
<td>4</td>
<td>0</td>
<td>2.6</td>
</tr>
<tr>
<td>4m(U)(12)</td>
<td>1.5</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>8m(S)(19)</td>
<td>33</td>
<td>20</td>
<td>12</td>
<td>28.6</td>
</tr>
<tr>
<td>8m(U)(19)</td>
<td>49.6</td>
<td>117</td>
<td>117</td>
<td>83.8</td>
</tr>
<tr>
<td>12m(S)(26)</td>
<td>101</td>
<td>1623</td>
<td>1611</td>
<td>191</td>
</tr>
<tr>
<td>Ex3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4d(S)(12)</td>
<td>5.8</td>
<td>10</td>
<td>4</td>
<td>131</td>
</tr>
<tr>
<td>4d(U)(12)</td>
<td>55.6</td>
<td>188</td>
<td>126</td>
<td>126.1</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Ex (S/U)</th>
<th># mult+</th>
<th>δ</th>
<th>CORD</th>
<th>iSAT [14]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>div</td>
<td></td>
<td></td>
<td>n</td>
</tr>
<tr>
<td>e1(U)</td>
<td>1+0</td>
<td>10</td>
<td>12</td>
<td>U</td>
</tr>
<tr>
<td>e2(S)</td>
<td>1+0</td>
<td>13</td>
<td>15</td>
<td>S</td>
</tr>
<tr>
<td>e3(U)</td>
<td>1+0</td>
<td>15</td>
<td>17</td>
<td>U</td>
</tr>
<tr>
<td>e4(S)</td>
<td>1+0</td>
<td>20</td>
<td>22</td>
<td>S</td>
</tr>
</tbody>
</table>

### NS: Checking numerical stability

<table>
<thead>
<tr>
<th>Ex (S/U)</th>
<th># mult+</th>
<th>δ</th>
<th>CORD</th>
<th>iSAT [14]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>div</td>
<td></td>
<td></td>
<td>n</td>
</tr>
<tr>
<td>s1(U)</td>
<td>1+0</td>
<td>18</td>
<td>20</td>
<td>U</td>
</tr>
<tr>
<td>s2(U)</td>
<td>1+0</td>
<td>18</td>
<td>20</td>
<td>U</td>
</tr>
<tr>
<td>s3(S)</td>
<td>1+0</td>
<td>18</td>
<td>20</td>
<td>S</td>
</tr>
<tr>
<td>s4(S)</td>
<td>1+1</td>
<td>18</td>
<td>21</td>
<td>S</td>
</tr>
</tbody>
</table>

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.