Newton’s method extended to $\omega$-continuous semi-rings

Prathmesh Prabhu
Taedong Kim
Srikrishna Sridhar
Project Goal

• Try out Newton’s method extended to semi-rings to solve program analysis problems\(^1\).

\(^1\) Esparza et. al “Newton’s method on ω continuous semi-rings”
Outline

• Background & Motivation
• Approach
  • Past Work
  • An alternate approach
• Results
  • Discussion
  • Future directions
BACKGROUND & MOTIVATION
Problem Statement

Compute the least solution of a system of equations defined on an $\omega$–continuous semi-ring.

$$\vec{x} = \vec{F}(\vec{x})$$
A semi-ring is an algebraic structure: \((S, \oplus, \otimes)\)

\(\oplus, \otimes\) are internal compositions on \(S\) with the following properties -

\[
(a \oplus b) \oplus c = a \oplus (b \oplus c) \quad (\oplus \text{ assoc.})
\]

\[
a \oplus b = b \oplus a \quad (\oplus \text{ commutative})
\]

\[
0 \oplus a = a = a \oplus 0 \quad (\text{additive identity})
\]

\[
(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)
\]

\[
0 \otimes a = 0 = a \otimes 0 \quad (0 \text{ annihilates } \otimes)
\]

\[
a \oplus a = a \quad (\text{optional idempotency})
\]
Semi-rings – Intuition

\((S, \otimes, \oplus)\)
Semi-rings – Intuition

\((S, \otimes, \oplus)\)

• Let’s take \(\mathbb{R}\)

\[\oplus : + \quad \otimes : \times\]
Semi-rings – Intuition

\((S, \otimes, \oplus)\)

- Let’s take \(\mathbb{R}\)
  
  \[
  \begin{align*}
  \oplus &: + \\
  \otimes &: \times
  \end{align*}
  
- \(\otimes\) not commutative
Semi-rings – Intuition

\((S, \otimes, \oplus)\)

• Let’s take \(\mathbb{R}\)
  
  \(\oplus : +\) \hspace{2cm} \(\otimes : \times\)

• \(\otimes\) not commutative
• No ‘-’ operation
Semi-rings – Intuition

\((S, \otimes, \oplus)\)

- Let’s take \(\mathbb{R}\)
  
  \(\oplus : +\)
  \(\otimes : \times\)

- \(\otimes\) not commutative
- No ‘-’ operation
- No ‘/’ operation
Semi-rings vis-à-vis Lattices

1. Join semilattices are the same as semi-rings -

   \[ \lor : \oplus \]
Semi-rings vis-à-vis Lattices

1. Join semilattices are the same as semi-rings -

\[ \lor : \oplus \]

1. Program semantics
Semi-rings vis-à-vis Lattices

1. Join semilattices are the same as semi-rings -

   \[ \lor : \oplus \]

   1. Program semantics
   2. Abstract interpretation
Semi-rings vis-à-vis Lattices

1. Join semilattices are the same as semi-rings -

   \[ \lor : \oplus \]

2. Many useful monotonic functions that arise in program analysis can be written using \[ \otimes \]

1. Program semantics
2. Abstract interpretation
Semi-rings are useful!

Some well studied problems:
Semi-rings are useful!

Some well studied problems:

• Reachability Analysis
Semi-rings are useful!

Some well studied problems:

• Reachability Analysis

• Boolean Relational Analysis
Semi-rings are useful!

Some well studied problems:

- Reachability Analysis
- Boolean Relational Analysis
- Affine Relations Analysis (ARA)
Write a system of equations, whose fixed point solution is exactly the set of reachable nodes in this CFG:
Write a system of equations, whose fixed point solution is exactly the set of reachable nodes in this CFG:

\[ S = \{ r, n \} \]
Write a system of equations, whose fixed point solution is exactly the set of reachable nodes in this CFG:

\[ S = \{ r, n \} \]

Variables: p1, p2, p3, p4, q1, q2
The reachability toy example

Write a system of equations, whose fixed point solution is exactly the set of reachable nodes in this CFG:

\[ S = \{ r, n \} \]

Variables: p1, p2, p3, p4, q1, q2

Initial Values:
\[ p1 = r \]
\[ p2, p3, p4, q1, q2 = n \]
Write a system of equations, whose fixed point solution is exactly the set of reachable nodes in this CFG:

\[ S = \{ r, n \} \]

Variables: \( p_1, p_2, p_3, p_4, q_1, q_2 \)

Initial Values:
\[ p_1 = r \]
\[ p_2, p_3, p_4, q_1, q_2 = n \]
The reachability toy example

Write a system of equations, whose fixed point solution is exactly the set of reachable nodes in this CFG:

Variables: p1, p2, p3, p4, q1, q2

Initial Values:
\[ p1 = r \]
\[ p2, p3, p4, q1, q2 = n \]

\[ p2 = p1 \otimes r \]
Write a system of equations, whose fixed point solution is exactly the set of reachable nodes in this CFG:

\[ S = \{ r, n \} \]

Variables: \( p_1, p_2, p_3, p_4, q_1, q_2 \)

Initial Values:
\[ p_1 = r \]
\[ p_2, p_3, p_4, q_1, q_2 = n \]

\[ p_2 = p_1 \otimes r \]
\[ q_1 = p_2 \otimes r \]
\[ q_2 = q_1 \otimes r \]
\[ p_3 = p_2 \otimes q_2 \]
\[ p_4 = p_3 \otimes r \]
Past and Present

APPROACH
How do we solve such systems?

\[ p_2 = p_1 \otimes r \]
\[ q_1 = p_2 \otimes r \]
\[ q_2 = q_1 \otimes r \]
\[ p_3 = p_2 \otimes q_2 \]
\[ p_4 = p_3 \otimes r \]
How do we solve such systems?

\[ p_2 = p_1 \otimes r \]
\[ q_1 = p_2 \otimes r \]
\[ q_2 = q_1 \otimes r \]
\[ p_3 = p_2 \otimes q_2 \]
\[ p_4 = p_3 \otimes r \]

\[ \vec{x} = \mathbb{F}(\vec{x}) \]
Solving $\vec{x} = F(\vec{x})$

Some useful properties
Solving $\vec{x} = \mathbb{F}(\vec{x})$

Some useful properties

- Domain of $X$ (semi-ring) is a partial order.
Solving \[ \vec{x} = \mathbb{F}(\vec{x}) \]

Some useful properties

- Domain of \( X \) (semi-ring) is a partial order.
- \( F \) is continuous and monotonic.
Solving $\vec{x} = F(\vec{x})$

Some useful properties

- Domain of $X$ (semi-ring) is a partial order.
- $F$ is continuous and monotonic.

Hence, we can apply a technique called Kleene’s Method.
Kleene’s Iteration Scheme

- Start from bottom and recursively apply $F$

\[ \bot \leq f(\bot) \leq f(f(\bot)) \leq \ldots \leq f^n(\bot) \leq \ldots \]
Kleene’s Iteration Scheme

\[ f^n(\perp) \]

\[ f(f(\perp)) \]

\[ f(\perp) \]
Kleene’s Iteration Scheme

• Start from bottom and recursively apply \( F \)

\[
\bot \leq f(\bot) \leq f(f(\bot)) \leq \ldots \leq f^n(\bot) \leq \ldots
\]

• Series approaches least fixed point(lfp).

\[
lfp(f) = \sup(f^n(\bot) \mid n \in \mathbb{N})
\]
Kleene’s Iteration Scheme

\[ k=0, \quad x_k = \bot \]

\[ x_{k+1} = F(x_k) \]

\[ x_{k+1} = x_k \]

- yes: \( x_k \) is the solution
- no: \( k = k + 1 \)
Drawbacks

- Kleene’s iteration scheme converges very slowly.
Another approach..

• Recently, Newton’s method has been extended to functions on semi-ring domains. ¹

¹ Esparza et. al “Newton’s method on ω continuous semi-rings”
Another approach..

• Recently, Newton’s method has been extended to functions on semi-ring domains.

Intuition

• In real domain

\[ x = F(x) \iff x - F(x) = 0 \]
If it’s fast, it’s Newton

• The Newton’s method is the most common technique to solve a root-finding problem.
If it’s fast, it’s Newton

• The Newton’s method is the most common technique to solve a root-finding problem.

• Quadratic convergence
  – Error term reduces quadratically.

\[
\frac{||x_{k+1} - x^*||}{||x_k - x^*||^2} \leq r
\]
If it’s fast, it’s Newton

- The Newton’s method is the most common technique to solve a root-finding problem.
- Quadratic convergence
  - Error term reduces quadratically.
    $$\frac{||x_{k+1} - x^*||}{||x_k - x^*||^2} \leq r$$

- Newton’s methods is robust on semi-rings.
Newton’s method

- Start with an initial guess of root of the function.
- Compute a linear approximation at the point.
- The x-intercept of the new point is a better approximation to the root.

\[ x_{n+1} = x_n - \frac{f'(x_n)}{f(x_n)} \]
Newton’s method

- Start with an initial guess of root of the function.
- Compute a linear approximation at the point.
- The x-intercept of the new point is a better approximation to the root.

\[ x_{n+1} = x_n - \frac{f'(x_n)}{f(x_n)} \]
Newton’s method

• As we saw, there is no subtraction (-) and division (/) on the semi-rings.
• Esparza et al. redefines linearization on semi-rings.

\[ x_{n+1} = x_n - \frac{f'(x_n)}{f(x_n)} \]
Linearization of \( f(X) = X \) at \( X = X_0 \)

is defined as \( Df_{X_0}(X) + f(X_0) = X \).
Linearization

Linearization of \( f(X) = X \) at \( X = X_0 \)

is defined as \( Df_{X_0}(X) + f(X_0) = X \).

Linearizing \( p_2 \otimes q_2 = p_3 \) at \( (p_2, q_2) = (n, n) \)?
Linearization

Linearization of \( f(X) = X \) at \( X = X_0 \)

is defined as \( Df_{X_0}(X) + f(X_0) = X \).

Linearizing \( p_2 \otimes q_2 = p_3 \) at \( (p_2, q_2) = (n, n) \)?

\[
(n \otimes q_2) \oplus (p_2 \otimes n) \oplus (n \otimes n) = p_3
\]
Linearization

Linearization of $f(X) = X$ at $X = X_0$

is defined as $Df_{X_0}(X) + f(X_0) = X$.

Linearizing $p_2 \otimes q_2 = p_3$ at $(p_2, q_2) = (n, n)$?

$$
\underbrace{(n \otimes q_2) \oplus (p_2 \otimes n)}_{Df_{X_0}(X)} \oplus \underbrace{(n \otimes n)}_{f(X_0)} = p_3
$$
Newton’s Iteration Scheme

1. \( k=0, x_k = \_ \)
2. Linearized \( F(x) \) at \( x_k \)
   \( F_k(x) \)
3. Solve \( x_{k+1} = F_k(x_{k+1}) \)
4. \( k = k+1 \)
5. If \( x_{k+1} = x_k \) then yes; \( x_k \) is the solution.
   Otherwise, no.
Newton’s Iteration Scheme

\[ k=0, \quad x_k = \_] \]

Linearized \( F(x) \) at \( x_k \)
\[ F_k(x) \]

Solve \( x_{k+1} = F_k(x_{k+1}) \)

Kleene’s method

\[ k = k+1 \]

\[ x_{k+1} = x_k \]

\[ x_k \text{ is the solution} \]
RESULTS & FUTURE DIRECTIONS
Experimental Results

• Verify correctness of the solvers for
  – Reachability Analysis
  – Boolean relational Analysis
Experimental Results

• Verify correctness of the solvers for
  – Reachability Analysis
  – Boolean relational Analysis

• Performance analysis for
  – Affine Relational Analysis
ARA Semi-Rings

• Test on Windows binaries of varying size.
ARA Semi-Rings

• Test on Windows binaries of varying size.

• Solve system of equations on Control flow graphs with transitions labeled with ARA weights.
• Test on Windows binaries of varying size.

• Solve system of equations on Control flow graphs with transitions labeled with ARA weights.

• Solve this system using Kleene and Newton methods.
Benchmarks

• Number of Kleene iterations
  – Note: For newton method, we add up Kleene iterations from all the Newton steps.

• Time taken
Time analysis

Kleene Iterations

Time Taken (s)

write (250)
Time analysis

Kleene Iterations

<table>
<thead>
<tr>
<th></th>
<th>Kleene</th>
<th>Newton</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>486</td>
<td>344</td>
</tr>
</tbody>
</table>

Time Taken (s)

<table>
<thead>
<tr>
<th></th>
<th>Kleene</th>
<th>Newton</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8.767</td>
<td>8.393</td>
</tr>
</tbody>
</table>

write (250)
Keep it Simple Stupid!

\[ p_3 = (n \otimes q^2) \oplus (p_2 \otimes n) \oplus (n \otimes n) \]
Keep it Simple Stupid!

\[ p_3 = (n \otimes q^2) \oplus (p_2 \otimes n) \oplus (n \otimes n) \]

\[ p_3 = (n \otimes q^2) \oplus p_3 \]

\[ p_3 = (p_2 \otimes n) \oplus p_3 \]

Starting from \( p_3 = n \otimes n \)
Keep it Simple Stupid!

\[ P_0 = f_{P_1}(P_1) + f_{P_2}(P_2) + f_{P_3}(P_3) \]
Keep it Simple Stupid!

\[ P_0 = f_{P_1}(P_1) + f_{P_2}(P_2) + f_{P_3}(P_3) \]

\[ P_0 = f_{P_1}(P_1) + P_0 \]
\[ P_0 = f_{P_2}(P_2) + P_0 \]
\[ P_0 = f_{P_3}(P_3) + P_0 \]
## Results Summary

<table>
<thead>
<tr>
<th>Problem Size</th>
<th>Kleene Solver</th>
<th>Newton</th>
<th>Modified Newton</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of iter.</td>
<td>Time (s)</td>
<td>Newton Iter.</td>
</tr>
<tr>
<td><strong>write</strong></td>
<td>250</td>
<td>486</td>
<td>8.767</td>
</tr>
<tr>
<td><strong>finger</strong></td>
<td>551</td>
<td>1579</td>
<td>73.274</td>
</tr>
<tr>
<td><strong>chkdsk</strong></td>
<td>1516</td>
<td>4322</td>
<td>103.319</td>
</tr>
<tr>
<td><strong>label</strong></td>
<td>1111</td>
<td>3989</td>
<td>144.488</td>
</tr>
<tr>
<td><strong>convert</strong></td>
<td>1952</td>
<td>4937</td>
<td>211.318</td>
</tr>
<tr>
<td><strong>comp</strong></td>
<td>2443</td>
<td>9889</td>
<td>790.578</td>
</tr>
<tr>
<td><strong>logoff</strong></td>
<td>2144</td>
<td>7769</td>
<td>1656.71</td>
</tr>
<tr>
<td><strong>route</strong></td>
<td>1738</td>
<td>6640</td>
<td>1228.81</td>
</tr>
<tr>
<td><strong>setup</strong></td>
<td>3463</td>
<td>9684</td>
<td>2817.05</td>
</tr>
<tr>
<td>Problem Size</td>
<td>Kleene Solver</td>
<td>Newton</td>
<td>Modified Newton</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>--------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td># of iter.</td>
<td>Time (s)</td>
<td># of iter.</td>
</tr>
<tr>
<td>write</td>
<td>250</td>
<td>486</td>
<td>8.767</td>
</tr>
<tr>
<td>finger</td>
<td>551</td>
<td>1579</td>
<td>73.274</td>
</tr>
<tr>
<td>chkdsk</td>
<td>1516</td>
<td>4322</td>
<td>103.319</td>
</tr>
<tr>
<td>label</td>
<td>1111</td>
<td>3989</td>
<td>144.488</td>
</tr>
<tr>
<td>convert</td>
<td>1952</td>
<td>4937</td>
<td>211.318</td>
</tr>
<tr>
<td>comp</td>
<td>2443</td>
<td>9889</td>
<td>790.578</td>
</tr>
<tr>
<td>logoff</td>
<td>2144</td>
<td>7769</td>
<td>1656.71</td>
</tr>
<tr>
<td>route</td>
<td>1738</td>
<td>6640</td>
<td>1228.81</td>
</tr>
<tr>
<td>setup</td>
<td>3463</td>
<td>9684</td>
<td>2817.05</td>
</tr>
</tbody>
</table>
Observations

- Number of newton steps is small.

- Low overhead between successive newton steps.
  - 98% time spent on solving linear systems
Future Directions

- Smarter Linear system solvers?
  - Incremental solvers.
  - Solvers that use the fact that systems are linear.
Thank you

Questions?

• Known unknowns

• Unknown unknowns