Uniformity Analysis for Irreducible CFGs

Sameer Sahasrabuddhe
Nicolai Hähnle
Motivation

• Support for irreducible CFGs
  • Existing Divergence Analysis conservatively assumes that all values are non-uniform.

• Implementation for Machine IR
  • Existing DA is implemented only on LLVM IR.
  • Uniformity Analysis is a template that works for LLVM IR and Machine IR.
Overview

• Uniformity of values is closely related to thread convergence.
• Start with a definition of convergent execution:
  • **Static and Dynamic Instances** of Operations.
  • Dynamic instances related by converged-with.
  • **Maximal Convergence**: A converged-with relation suitable for known use-cases.
• **m-converged** Static Instances
  • Static property suitable for irreducible CFGs.
  • Derived from the converged-with relation over dynamic instances.
• **Uniformity** defined using m-converged Static Instances.
References

The Uniformity Analysis extends the following work by introducing a conservative treatment of irreducible control flow graphs.

  
  https://doi.org/10.1145/3434312
Convergence and Uniformity

- Conventional picture:
  - Threads are **converged** until they **diverge** at a **divergent** branch.
  - Diverged threads eventually **reconverge** at some common program point.
  - **Convergent operations** require certain threads to execute them convergently.
- Convergently executed operations produce uniform values (**conditions apply**).
- A value computed by different threads is **uniform** if it is the same across those threads.
  - The value is **divergent** otherwise.
- A branch is **uniform** or **divergent** if its condition is uniform or divergent, respectively.

<table>
<thead>
<tr>
<th>Object</th>
<th>Can be …</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread</td>
<td>Converged</td>
</tr>
<tr>
<td>Communication</td>
<td>Convergent</td>
</tr>
<tr>
<td>Value</td>
<td>Uniform</td>
</tr>
<tr>
<td>Branch</td>
<td>Uniform</td>
</tr>
</tbody>
</table>
**Dynamic Instances**

- **Static instance**: Each occurrence of an instruction in the program source.
  - E.g.: The nodes H, B, L, etc in the adjoining CFG.
- **Dynamic instance**: Each execution of a static instance by a thread.
  - E.g.: The entries H1, B1, H2, etc in the table below

<table>
<thead>
<tr>
<th>Thread1</th>
<th>Entry1</th>
<th>H1</th>
<th>B1</th>
<th>L1</th>
<th>H3</th>
<th>L3</th>
<th>H5</th>
<th>L5</th>
<th>Exit1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread2</td>
<td>Entry2</td>
<td>H2</td>
<td>L2</td>
<td>H4</td>
<td>B2</td>
<td>L4</td>
<td></td>
<td></td>
<td>Exit2</td>
</tr>
</tbody>
</table>

Convention: Dynamic instances are listed in the same column if and only if they are converged.
Convergence = \{ \textit{converged-with} \textit{and} \textit{convergence-before} \}

- **converged-with**
  - Relates dynamic instances of the same static instance produced by different threads.
  - Transitive symmetric relation.
  - **No single definition**: Choose an instance that reflects the execution on the target.

- **convergence-before**
  - Produced by \textit{converged} dynamic instances.
  - Relates other dynamic instances in the corresponding threads.
  - Transitive strict partial order.

<table>
<thead>
<tr>
<th>Thread1</th>
<th>Thread2</th>
<th>Thread3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Converged:**
  - Q1 and Q2.
  - S1 and S2.

- **Convergence order:**
  - P → Q2
  - Q1 → R
  - P → T
Maximal Convergence: An instance of converged-with

• **Expectation 1:** Threads *should* converge as often as possible:
  • At a convergent operation.
  • At the header of a cycle (generalization of a natural loop; can be irreducible).
  • At a post-dominator.

• **Expectation 2:** When threads enter a cycle:
  • Threads may divergently exit the cycle on different iterations.
  • All threads must finish that cycle before reconverging outside.

• Formally captured as **maximal convergence:**
  • Suitable for existing targets.
  • Compatible with the *convergent* attribute.
  • Works with irreducible CFGs.
Maximal Convergence (Informally)

- R and S are cycle headers.
  - Each execution of a header marks a new iteration of the cycle.
- P1 converged-with P2 but not with P4 (different iterations).
- S3 not converged-with S4 (different iterations).
Maximal Convergence (Formally)

- **P1 not converged-with P4:**
  - Header R2 precedes P4 in the thread but not convergence-before P1.
- **S3 not converged-with S4:**
  - Header R3 precedes S4 in the thread but not convergence-before S3.
- **Dynamic instances X1 and X2 are converged if and only if:**
  - for every cycle that contains static instance X, there is no dynamic instance H' of the header H such that
    - H' precedes X1 (respectively, X2) in the same thread, and,
    - H' is not convergence-before X2 (respectively, X1).
Convergence in Irreducible CFGs

- An irreducible CFG can be resolved into cycles in different ways.
- Each cycle hierarchy produces its own convergence.

**Case 1: A single cycle with header P**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entry1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>Q3</td>
<td>R3</td>
<td>S3</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thread2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entry2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td></td>
<td>Q4</td>
<td>R2</td>
<td>S4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Case 2: Nested cycles with headers R and S**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entry1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>R1</td>
<td>S1</td>
<td>P3</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thread2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entry2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td></td>
<td>Q4</td>
<td>R2</td>
<td>S4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
M-Converged Static Instances

- A static instance X is **m-converged** if and only if
  - Its dynamic instances are **converged in the same way in every cycle hierarchy**.

- For reducible CFGs:
  - Unique loop hierarchy.
  - All static instances are **m-converged**.

- For irreducible CFGs:
  - Identify certain static instances as **m-converged**.
  - Based on “closed paths” in the CFG, which are independent of cycles.
Uniformity for Irreducible CFGs

• If a static instance is not \emph{m-converged}, outputs are assumed to be \emph{divergent}.
• If a static instance \( X \) is \emph{m-converged}, then the outputs are \emph{uniform} if:
  • The semantics of the instruction specifies the output to be \emph{uniform}, OR
  • Each incoming value is uniform, AND
    • If \( X \) is a PHI node, then \emph{converged} threads choose the same incoming value.
Implementation

• RFC posted as review on the LLVM Phabricator website:
  • https://reviews.llvm.org/D130746

• The analysis is implemented as a template that can be instantiated for both LLVM IR and Machine IR.

• Current status:
  • Passes existing tests for divergence analysis
  • Passes new tests with irreducible control flow
  • Currently working on Machine IR tests
COPYRIGHT AND DISCLAIMER

©2022 Advanced Micro Devices, Inc. All rights reserved.

AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate releases, for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED 'AS IS.” AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.