Optimal Chain Rule Placement for Instruction Selection based on SSA Graphs

Stefan Schäfer, Bernhard Scholz
(stefans|scholz)@it.usyd.edu.au

School of IT, University of Sydney
Outline

- Related Work
- Motivation (Instruction Selection based on SSA Form)
- Chain Rule Placement
- Implementation
- Results
- Conclusion
Instruction Selection based on SSA Graphs (1)
Related Work (1)

- Tree Pattern Matching

C. Fraser, R. Henry, T. Proebsting
*BURG – Fast Optimal Instruction Selection and Tree Parsing*

- Works fine with trees (expressions)
Related Work (1)

- Tree Pattern Matching

C. Fraser, R. Henry, T. Proebsting

*BURG – Fast Optimal Instruction Selection and Tree Parsing*


- Works fine with trees (expressions)
- Problem: control flow graphs are usually directed acyclic graphs
Related Work (1)

• Tree Pattern Matching

C. Fraser, R. Henry, T. Proebsting

*BURG – Fast Optimal Instruction Selection and Tree Parsing*


  - Works fine with trees (expressions)
  - Problem: control flow graphs are usually directed acyclic graphs

• Code Selection for DAGs

M. A. Ertl, *Optimal Code Selection in DAGs, Proceedings of POPL 1999*
Related Work (1)

- **Tree Pattern Matching**

  C. Fraser, R. Henry, T. Proebsting

  *BURG – Fast Optimal Instruction Selection and Tree Parsing*


  - Works fine with trees (expressions)
  - Problem: control flow graphs are usually directed acyclic graphs

- **Code Selection for DAGs**


  - DAG-Matching is NP-complete.
Related Work (2)

- **Code Selection based on SSA Graphs**

  E. Eckstein, O. König, B. Scholz
  
  *Code Instruction Selection based on SSA Graphs*

  SCOPES 2003, Volume 2826 of *Lecture Notes on Computer Science*

  - Introduced a (heuristical) code selection techniques for DAGs
  - Cost-optimal derivation of a graph grammar for a given SSA graph
  - Chain rules used for type conversion, but **optimal placement unaddressed**
  - **optimal** means: cost-minimal for a given cost metric
Instruction Selection based on SSA Graphs (2)
Instruction Selection based on SSA Graphs (2)
Instruction Selection based on SSA Graphs (2)

\[
\begin{align*}
\text{reg} & := \text{add}(\text{reg}, \text{reg}) & [10.0, 1.0] \\
\text{sreg} & := \text{cast}(\text{sreg}) & [10.0, 1.0] \\
\text{reg} & := \text{sreg} & [10.0, 1.0] \\
\text{sreg} & := \text{reg} & [10.0, 1.0]
\end{align*}
\]
Instruction Selection based on SSA Graphs (2)

\[
\begin{align*}
\text{reg} & ::= \text{add}(\text{reg}, \text{reg}) \quad [10.0, 1.0] \\
\text{sreg} & ::= \text{cast}(\text{sreg}) \quad [10.0, 1.0] \\
\text{reg} & ::= \text{sreg} \quad [10.0, 1.0] \\
\text{sreg} & ::= \text{reg} \quad [10.0, 1.0]
\end{align*}
\]
Instruction Selection based on SSA Graphs (2)

\[
\begin{align*}
\text{reg} & ::= \text{add}(\text{reg}, \text{reg}) & [10.0, 1.0] \\
\text{sreg} & ::= \text{cast}(\text{sreg}) & [10.0, 1.0] \\
\text{reg} & ::= \text{sreg} & [10.0, 1.0] \\
\text{sreg} & ::= \text{reg} & [10.0, 1.0]
\end{align*}
\]

<table>
<thead>
<tr>
<th>strategy</th>
<th>time</th>
<th>space</th>
<th>trade-off 1:4</th>
</tr>
</thead>
<tbody>
<tr>
<td>def ( (b_1) )</td>
<td>30140</td>
<td>1</td>
<td>6028.8</td>
</tr>
</tbody>
</table>
Instruction Selection based on SSA Graphs (2)

\[
\begin{align*}
\text{reg} & := \text{add}(\text{reg}, \text{reg}) \\
\text{sreg} & := \text{cast}(\text{sreg}) \\
\text{reg} & := \text{sreg} \\
\text{sreg} & := \text{reg}
\end{align*}
\]

\[
\begin{align*}
\text{strategy} & \quad \text{time} & \quad \text{space} & \quad \text{trade-off 1:4} \\
\text{def} (b_1) & \quad 30140 & \quad 1 & \quad 6028.8 \\
\text{uses} (b_{11}, b_{12}, b_{14}) & \quad 19300 & \quad 3 & \quad 3862.4
\end{align*}
\]
Instruction Selection based on SSA Graphs (2)

```
reg ::= add(reg, reg) [10.0, 1.0]
sreg ::= cast(sreg) [10.0, 1.0]
reg ::= sreg [10.0, 1.0]
sreg ::= reg [10.0, 1.0]
```

<table>
<thead>
<tr>
<th>strategy</th>
<th>time</th>
<th>space</th>
<th>trade-off 1:4</th>
</tr>
</thead>
<tbody>
<tr>
<td>def ($b_1$)</td>
<td>30140</td>
<td>1</td>
<td>6028.8</td>
</tr>
<tr>
<td>uses ($b_{11}$, $b_{12}$, $b_{14}$)</td>
<td>19300</td>
<td>3</td>
<td>3862.4</td>
</tr>
<tr>
<td>def/uses</td>
<td>19300</td>
<td>1</td>
<td>3862.4</td>
</tr>
</tbody>
</table>
Instruction Selection based on SSA Graphs (2)

\[
\begin{align*}
\text{reg} & ::= \text{add} (\text{reg}, \text{reg}) & [10.0, 1.0] \\
\text{sreg} & ::= \text{cast} (\text{sreg}) & [10.0, 1.0] \\
\text{reg} & ::= \text{sreg} & [10.0, 1.0] \\
\text{sreg} & ::= \text{reg} & [10.0, 1.0]
\end{align*}
\]

<table>
<thead>
<tr>
<th>strategy</th>
<th>time</th>
<th>space</th>
<th>trade-off 1:4</th>
</tr>
</thead>
<tbody>
<tr>
<td>def ((b_1))</td>
<td>30140</td>
<td>1</td>
<td>6028.8</td>
</tr>
<tr>
<td>uses ((b_{11}, b_{12}, b_{14}))</td>
<td>19300</td>
<td>3</td>
<td>3862.4</td>
</tr>
<tr>
<td>def/uses</td>
<td>19300</td>
<td>1</td>
<td>3862.4</td>
</tr>
<tr>
<td>optimal</td>
<td>3510</td>
<td></td>
<td></td>
</tr>
<tr>
<td>placed at</td>
<td>(b, b, b)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Instruction Selection based on SSA Graphs (2)

\[
\begin{array}{c}
\text{def (} b_1 \text{)} \\
30140 \\
1 \\
6028.8 \\
\text{uses (} b_{11}, b_{12}, b_{14} \text{)} \\
19300 \\
3 \\
3862.4 \\
\text{def/uses} \\
19300 \\
1 \\
3862.4 \\
\text{optimal} \\
3510 \\
1 \\
\end{array}
\]

strategies:
- def (\(b_1\))
- uses (\(b_{11}, b_{12}, b_{14}\))
- def/uses
- optimal

placements:
- \(b_5\)
- \(b_9\)
- \(b_{10}\)
- \(b_{12}\)
- \(b_1\)
- \(b_2\)
- \(b_3\)
- \(b_6\)
- \(b_7\)
- \(b_8\)
- \(b_9\)
- \(b_{10}\)
- \(b_{11}\)
- \(b_{12}\)
- \(b_{14}\)

\[
\begin{array}{cccc}
\text{time} & \text{space} & \text{trade-off} 1:4 \\
30140 & 1 & 6028.8 \\
19300 & 3 & 3862.4 \\
19300 & 1 & 3862.4 \\
3510 & 1 & \\
\end{array}
\]

reg := add(reg, reg)  [10.0, 1.0]

sreg := cast(sreg)  [10.0, 1.0]

reg := sreg  [10.0, 1.0]

sreg := reg  [10.0, 1.0]
Instruction Selection based on SSA Graphs (2)

\[
\begin{align*}
\text{reg} & := \text{add} (\text{reg}, \text{reg}) & [10.0, 1.0] \\
\text{sreg} & := \text{cast} (\text{sreg}) & [10.0, 1.0] \\
\text{reg} & := \text{sreg} & [10.0, 1.0] \\
\text{sreg} & := \text{reg} & [10.0, 1.0]
\end{align*}
\]

<table>
<thead>
<tr>
<th>strategy</th>
<th>time</th>
<th>space</th>
<th>trade-off 1:4</th>
</tr>
</thead>
<tbody>
<tr>
<td>def ((b_1))</td>
<td>30140</td>
<td>1</td>
<td>6028.8</td>
</tr>
<tr>
<td>uses ((b_{11}, b_{12}, b_{14}))</td>
<td>19300</td>
<td>3</td>
<td>3862.4</td>
</tr>
<tr>
<td>def/uses</td>
<td>19300</td>
<td>1</td>
<td>3862.4</td>
</tr>
</tbody>
</table>

optimal 

placed at \(b, b, b\)

\[b\]

\[b, b\]
SSA Form

- Single Static Assignment form
- There is at most one assignment to each variable.
- Each definition of a variable is distinct.
SSA Form

- Single Static Assignment form
- There is at most one assignment to each variable.
- Each definition of a variable is distinct.
- Multiple definitions have to be resolved:
  - if (e) \( b = 32 \) else \( b = 42 \); \( \rightarrow \) if (e) \( b_1 = 32 \) else \( b_2 = 42 \);
- Further uses induce \( \phi \)-functions:
  - \( a = b; \rightarrow a = \phi(b_1, b_2); \)
- SSA graphs as intermediate data flow representation in SSA form
Chain Rule Placement

- Map the CFG to a network
- Reduce the network for each definition and non-terminal
  (a definition node dominates all of its users)
- Find a minimum cut for each reduced network
Mapping to a Network
Mapping to a Network
Reducing each Network

- Done for each definition $d$ and non-terminal
- Starts in each user $u$:

  - Case 1: $u$ is not a $\phi$-node
Reducing each Network

- Done for each definition $d$ and non-terminal
- Starts in each user $u$:
- Case 1: $u$ is not a $\phi$-node
  - All nodes an all acyclic paths from $d$ to $u$ are dominated by $d$
  - All those nodes added to reduced network
Reducing each Network

- Done for each definition $d$ and non-terminal
- Starts in each user $u$:
- Case 2: $u$ is a $\phi$-node, all $v \in \text{preds}(u)$ is dominated by $d$

```
\[ w_1 = \text{op}(...) \]
\[ w_2 = \text{op'}(...) \]
\[ v_1 \]
\[ v_2 \]
\[ u = \phi(\ldots, w_1, \ldots, w_2 \ldots) \]
```
Reducing each Network

- Done for each definition \( d \) and non-terminal
- Starts in each user \( u \):
  - Case 2: \( u \) is a \( \phi \)-node, all \( v \in \text{preds}(u) \) are dominated by \( d \)
    - All nodes an all acyclic paths from \( d \) to \( v \) are dominated by \( d \)
    - All those nodes and \( u \) added to reduced network

\[
\begin{align*}
\text{Case 2: } & u \text{ is a } \phi\text{-node, all } v \in \text{preds}(u) \text{ are dominated by } d \\
& \quad \text{All nodes an all acyclic paths from } d \text{ to } v \text{ are dominated by } d \\
& \quad \text{All those nodes and } u \text{ added to reduced network}
\end{align*}
\]
Reducing each Network

- Done for each definition $d$ and non-terminal
- Starts in each user $u$:
- Case 3: $u$ is a $\phi$-node, any $v \in \text{preds}(u)$ is not dominated by $d$

![Diagram]

\[
d_1 = \text{op} (...) \\
\]

\[
d_2 = \text{op}' (...) \\
\]

\[
x_1 \\
\]

\[
x_2 \\
\]

\[
y \\
\]

\[
u = \phi(\ldots, d_1, \ldots, d_2 \ldots)\]
Reducing each Network

- Done for each definition $d$ and non-terminal
- Starts in each user $u$:
- Case 3: $u$ is a $\phi$-node, any $v \in \text{preds}(u)$ is not dominated by $d$
  - Stop traversal for all users of $d$ and add only $d$ to reduced network

```
d_2 = op'(\ldots)

\begin{align*}
\phi(\ldots, d_1, \ldots, d_2 \ldots) &
\end{align*}
```
Reducing each Network

- Done for each definition $d$ and non-terminal
- Starts in each user $u$:
- Case 3: $u$ is a $\phi$-node, any $v \in \text{preds}(u)$ is not dominated by $d$
  - Stop traversal for all users of $d$ and add only $d$ to reduced network

not cost-optimal but

does not occur very often:

2264628 nodes
94183 $\phi$-uses
case 3 occurs 1076 times
Implementation
Implementation

Graph Grammar

Code Generator Generator

Source for Code Generator in $L$

Code Base in $L$
Implementation

Graph Grammar

Code Base in $L$

Code Generator Generator

Source for Code Generator in $L$

PBQP Library for $L$

Compiler for $L$

Code Generator in $L$
Implementation

Graph Grammar

Code Base in $L$

Code Generator Generator

Source for Code Generator in $L$

PBQP Library for $L$

Compiler for $L$

Chain Rule Placement

Base Rule Matching

Complete Matching

Complete Matching

Input Program in SSA Form

Run

Code Generator in $L$
Costs (Spec2000, Time:Space 1:4)
Costs (MiBench, Time:Space 1:4)
Execution Times (Spec2000)
Contributions

- Contributed to code selection based on SSA-Graphs
- Main Contributions:
  - Formally addressed the unsolved problem of placing chain rules optimally
  - Introduced an efficient and effective algorithm to place chain rules optimally with respect to an arbitrary cost metric
  - Implemented a free, open-source code generator, enhancing rule matching with chain rule placement
  - Proved the correctness of our algorithm
  - Conducted experiments with Spec2000 and MiBench suites
Thank you for your attention!

Any questions or comments?