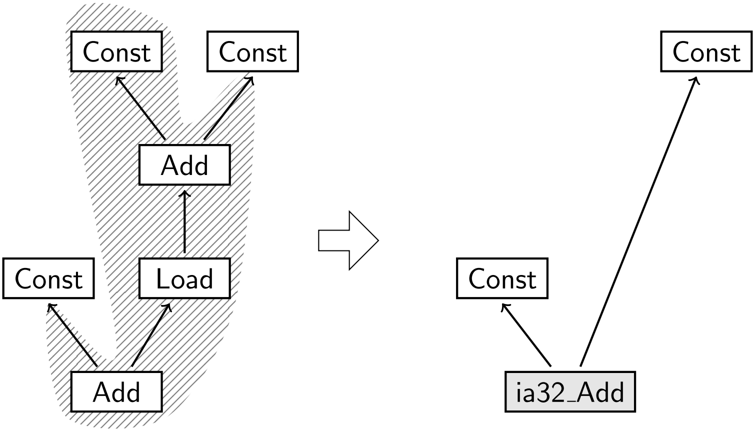
Instruction Selection on SSA Graphs

Sebastian Hack, Sebastian Buchwald, Andreas Zwinkau

Compiler Construction Course
Winter Term 2009/2010

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Instruction Selection



Instruction Selection on SSA

- "Optimal" instruction selection on trees is polynomial
- SSA programs are directed graphs
  $\Longrightarrow$ Data dependence graphs
- Translating back from SSA graphs to trees is not satisfactory
- "Optimal" instruction selection is NP-complete on DAGs
- The problem is common subexpressions
- Doing it on graphs provides more opportunities for complex instructions:
  - Patterns with multiple results
  - DAG-like patterns

Instruction Selection on SSA

- Graph Rewriting
- For every machine instruction specify:
  - A set of graphs (patterns) of IR nodes
  - Every pattern has associated costs

1. Find all matchings of the patterns in the IR graph
2. Pick a correct and optimal matching
3. Replace each pattern by corresponding machine instruction

$\Longrightarrow$ Result is an SSA graph with machine nodes

Graphs

- Let $G = (V, E)$ be a directed acyclic graph (DAG)
- Let Op be a set of operators
- Every node has a degree $\deg v : V \to \mathbb{N}_0$
- Every node $v \in V$ has an operator: $\mathrm{op} : V \to Op$
- Every operator $o \in Op$ has an arity $\# : Op \to \mathbb{N}_0$
- Let $\square \in Op$ be an operator with $\# \square = 0$
- Nodes with operator $\square$ denote "glue" points in the patterns (later)
- Every node's degree must match the operator's arity:

$$\# \, \mathrm{op} \, v = \deg v$$

Definition (Program Graph)

A graph G is a program graph if it is acyclic and

$$\forall v \in V : \mathrm{op} \, v \neq \square$$

Patterns

- A graph $P = (V, E)$ is rooted if there exists a node $v \in V_P$ such that there is a path from v to every node v' in P
- If P is a DAG, then there is a unique root called $\operatorname{rt} P$

Definition (Pattern Graph, Pattern)

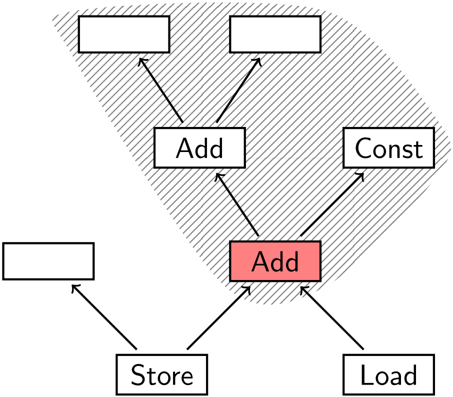
A graph P is a pattern if

- it is acyclic and rooted
- $\operatorname{op} \operatorname{rt} P \neq \square$

- Note that we explicitly allow nodes with operator $\square$ in patterns

Equivalence of Nodes in Patterns

- Complex patterns often have common sub-patterns



- Shall be treated as equivalent
- Selecting the common sub-pattern at the Add node shall enable selecting the complex instruction at Store and Load

Equivalence of Nodes in Patterns

Definition (Equivalence of nodes)

Consider two patterns P and Q and two nodes $v \in P$, $w \in Q$:

$$\begin{aligned} v \sim w :\iff\ & v = w \\ & \vee (\operatorname{span} v \cong \operatorname{span} w \wedge \operatorname{rt} P \neq v \wedge \operatorname{rt} Q \neq w) \end{aligned}$$

- Either the two nodes are identical
- v, w are no pattern roots and their spanned subgraphs are isomorphic
- $\operatorname{span} v$: induced subgraph that contains all nodes reachable from v

Matching of a Node

- Let $\mathcal{P} = \{P_1, P_2, \dots\}$ be a set of patterns
- Let G be some program graph

Definition (Matching)

A matching $\mathcal{M}_v$ of a node $v \in V_G$ with a set of patterns $\mathcal{P}$ is a family of pairs

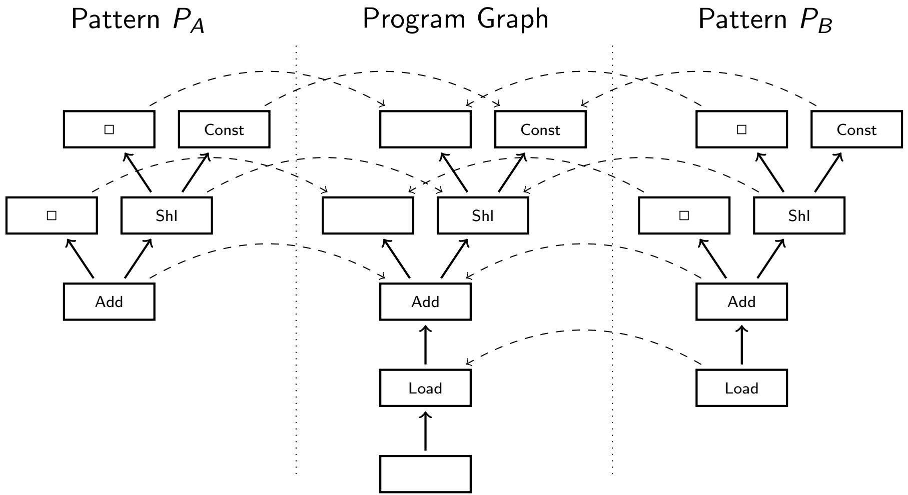
$$\mathcal{M}_v = \big((P_i, \imath_i)\big)_{i \in I} \qquad I \subseteq \{1, \dots, |\mathcal{P}|\}$$

of patterns and injective graph morphisms $\imath_i : P_i \to G$ satisfying

$$v \in \operatorname{ran} \imath_i \quad \text{and} \quad \operatorname{op} w \neq \square \implies \operatorname{op} w = \operatorname{op} \imath_i(w) \qquad \forall w \in P_i$$

Matchings

Example



Selection

- We have computed a covering of the graph
- i.e. instruction selection possibilities
- Now, find a subset of the covering that leads to good and correct code
- Cast the problem as a mathematical optimization problem:

Partitioned Boolean Quadratic Programming (PBQP)

PBQP

Let $\mathbb{R}_\infty = \mathbb{R}_+ \cup \{\infty\}$ and

- $\vec{c}_i \in \mathbb{R}_\infty^{k_i}$ be cost vectors
- $C_{ij} \in \mathbb{R}_\infty^{k_i} \times \mathbb{R}_\infty^{k_j}$ be cost matrices

Definition (PBQP)

Minimize

$$\sum_{1 \le i < j \le n} \vec{x}_i^\top \cdot C_{ij} \cdot \vec{x}_j + \sum_{1 \le i \le n} \vec{x}_i^\top \cdot \vec{c}_i$$

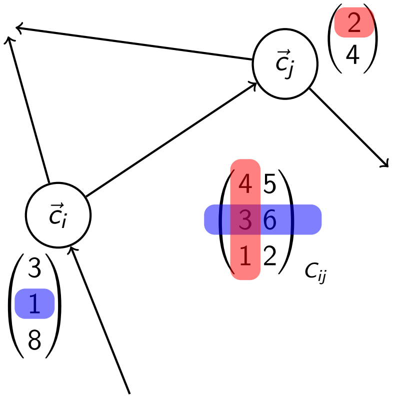
with respect to

$$\begin{aligned} &\vec{x}_i \in \{0, 1\}^{k_i} \\ &\vec{x}_i^\top \cdot \vec{1} = 1, \quad 1 \le i \le n \\ &\vec{x}_i^\top \cdot C_{ij} \cdot \vec{x}_j < \infty, \quad 1 \le i < j \le n \end{aligned}$$

PBQP

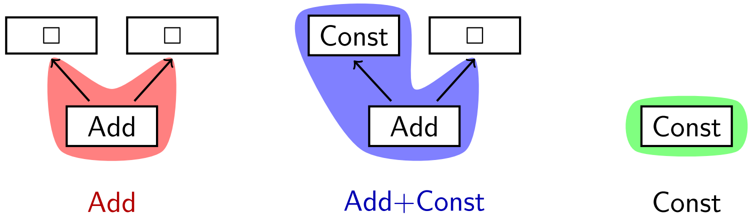
- $\vec{x}_i$ are selection vectors
- Exact one component is 1
- This selects the component
- Cost matrices relate selection of made in different selection vectors
- Can be modelled as a graph:
  - cost vectors are nodes
  - matrices are edges
  - only draw non-null matrix edges

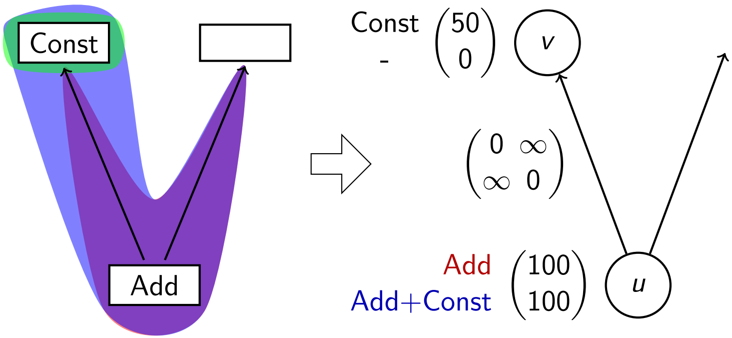
PBQP as a Graph



- Colors indicate selection vectors $\vec{x}_i = (0\,1\,0)^\top$ and $\vec{x}_j = (1\,0)^\top$
- This selection contribute the cost of 6 to the global costs
- Edge direction solely to indicate order of ij in the matrix subscript

Mapping Instruction Selection to PBQP





Mapping Instruction Selection to PBQP

Cost vectors are defined by node coverings:

- Let $\mathcal{C}_v$ be a node covering of v
- The alternatives of the node are given by partitioning the coverings by equivalence:

$$\mathcal{C}_{v/\sim}$$

- Common sub-patterns have to result in the same choice
- Costs come from an external specification

Mapping Instruction Selection to PBQP

- Matrices have to maintain selection correctness
- Consider two alternatives

$$A_u = (P_u, \imath_u) \quad A_v = (P_v, \imath_v)$$

  at two nodes u, v

- The matrix entry for those alternatives is

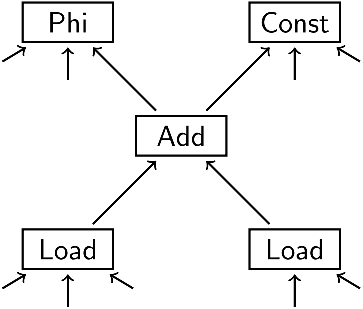
$$c(A_u, A_v) = \begin{cases} \infty & \text{op}\, \imath_u^{-1}(v) = \square \text{ and } \imath_v^{-1}(v) \neq \text{rt}\, P_v \\ \infty & \text{op}\, \imath_u^{-1}(v) \neq \square \text{ and } \imath_u^{-1}(v) \not\sim \imath_v^{-1}(v) \\ 0 & \text{else} \end{cases}$$

Id est:

- If A_u selects a leaf at v, A_v has to select a root
- If A_u does not select a leaf, both subpatters have to be equivalent

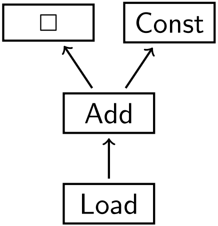
Example

Program Graph

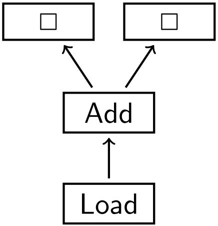


Example

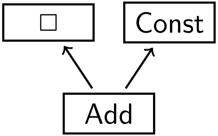
Patterns



LAC (Load+Add+Const)



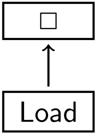
LA (Load+Add)



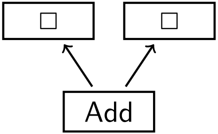
AC (Add+Const)

C (Const)

P (Phi)



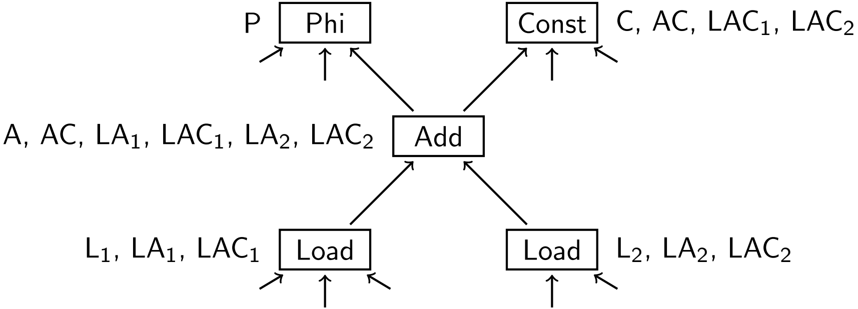
L (Load)



A (Add)

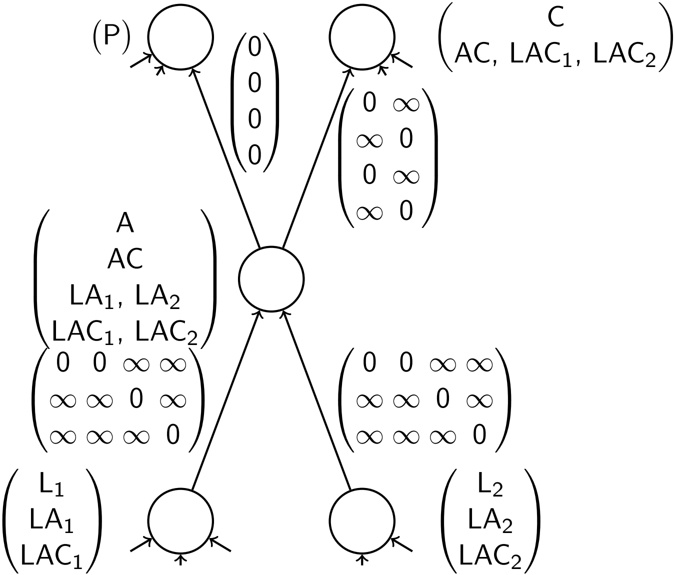
Example

Matchings



Example

PBQP Instance



Reducing the Problem

Optimality-preserving reductions of the problem:

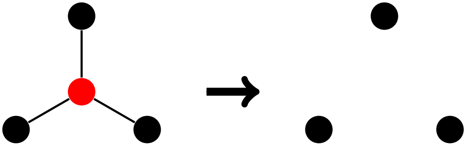
- Independent edges (e.g. matrix of zeroes):

- Nodes of degree 1:

- Nodes of degree 2:

Reducing the Problem

- Heuristic Reduction:



Chose the local minimum at a node

- Leads to a linear algorithm
- Each reduction eliminates at least one edge
- If all edges are reduced, minimizing nodes separately is easy

Summary

- Map instruction selection to an optimization problem
- SSA graphs are sparse $\implies$ reductions often applied
- In practice: heuristic reduction rarely happens
- Efficiently solvable
- Convenient mechanism:
  - Implementor specifies patterns and costs
  - maps each pattern to an machine node
  - Rest is automatic

- Criteria for pattern sets that allow for correct selections in every program not discussed here!

Literature

Sebastian Buchwald and Andreas Zwinkau.
Befehlsauswahl auf expliziten Abhängigkeitsgraphen.
Master's thesis, Universität Karlsruhe (TH), Dec 2008.

Erik Eckstein, Oliver König, and Bernhard Scholz.
Code Instruction Selection Based on SSA-Graphs.
In *SCOPES*, pages 49–65, 2003.

Hannes Jakschitsch.
Befehlsauswahl auf SSA-Graphen.
Master's thesis, Universität Karlsruhe, November 2004.