The Complexity of Register Allocation

Philipp Klaus Krause

October 29, 2011
<table>
<thead>
<tr>
<th></th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Register Allocation</td>
</tr>
<tr>
<td>2</td>
<td>Graph Coloring</td>
</tr>
<tr>
<td>3</td>
<td>Hardness</td>
</tr>
</tbody>
</table>
Table of Contents

1. Register Allocation

2. Graph Coloring

3. Hardness
Register Allocation

- Programs use variables that have to be stored somewhere.
- Compiler decides where to store variables.
  - Registers vs. memory.
  - Very important for code quality.
int binomial_coefficient(int n, int k) {
    int i, delta, max, c;
    if(n < k)
        return(0);
    if(n == k)
        return(1);
    if(k < n - k) {
        delta = n - k;
        max = k;
    } else {
        delta = k;
        max = n - k;
    }
    c = delta + 1;
    for(i = 2; i <= max; i++)
        c = (c * (delta + i)) / i;
    return(c);
}
int binomial_coefficient(int n, int k) {
    int i, delta, max, c;
    if(n < k)
        return(0);
    if(n == k)
        return(1);
    if(k < n - k) {
        delta = n - k;
        max = k;
    } else {
        delta = k;
        max = n - k;
    }
    c = delta + 1;
    for(i = 2; i <= max; i++)
        c = (c * (delta + i)) / i;
    return(c);
}
int binomial_coefficient(int n, int k)
{
    int i, delta, max, c;
    if(n < k)
        return(0);
    if(n == k)
        return(1);
    if(k < n - k)
    {
        delta = n - k;
        max = k;
    }
    else
    {
        delta = k;
        max = n - k;
    }
    c = delta + 1;
    for(i = 2; i <= max; i++)
        c = (c * (delta + i)) / i;
    return(c);
}
```c
int binomial_coefficient(int n, int k) {
    int i, delta, max, c;
    if(n < k)
        return(0);
    if(n == k)
        return(1);
    if(k < n - k) {
        delta = n - k;
        max = k;
    } else {
        delta = k;
        max = n - k;
    }
    c = delta + 1;
    for(i = 2; i <= max; i++)
        c = (c * (delta + i)) / i;
    return(c);
}
```
Given an input program and parameter $r$, the number of registers and a number $g$, the problem of register allocation is to decide if there is an $r$-colorable induced subgraph $S$ in the conflict graph, such that the sum $\sum_{v \in V \setminus V(S)} c(v)$ of the costs of the variables outside this subgraph is at most $g$. 
Assuming all $r$ registers are equal:

- Registers are colors.
- Color the conflict graph.
- Placing as many variables in registers as possible is equivalent to finding a maximum $r$-colorable induced subgraph.

---

1G. Chaitin, 1982
Problems with Graph Coloring

- NP-hard:
  Approximations are used, leading to suboptimal colorings.

- Hard to generalize:
  Register preferences, aliasing, etc are hard to model.

- Solution to NP-Hardness: Restricting the class of input graphs.
Many problems that are NP-hard can be solved efficiently when restricted to graphs of bounded tree-width.

The tree-width of control-flow graphs is bounded\(^2\).

For structured programs, and fixed \(r\) the decision problem of \(r\)-colorability of the conflict graph can be solved in linear time\(^3\).

For structured programs, and fixed \(r\) the register allocation problem can be solved in polynomial time\(^4\).

\(^2\)M. Thorup, 1998

\(^3\)H. Bodlaender et alii, 1998

\(^4\)K.
Table of Contents

1 Register Allocation
2 Graph Coloring
3 Hardness
Given a circuit $C$ (input) and an integer $k$ (parameter) the weighted circuit satisfiability problem asks if there is a satisfying assignment for $C$ of weight exactly $k$ (i.e. exactly $k$ of the inputs are set to “true” in the assignment).
Parametrized Complexity Classes

- \textsf{fpt} is the class of all parametrized problems parametrized by $k$ that can be solved in time $f(k)p(n)$ for input size $n$, computable $f$ and polynomial $p$.
- $W[t]$ is the class of all parametrized problems that can be $\textsf{fpt}$-reduced to the WCS problem on weft $t$, depth $d$ SAT-circuits, for a constant $d \geq 1$.
- $W[\text{SAT}]$ is the class of all parametrized problems that are fixed-parameter reducible to the WCS problem on SAT-circuits.
- $W[P]$ is the class of all parametrized problems that are fixed-parameter reducible to the WCS problem.
- $XP$ is the class of all problems parametrized by $k$ that can be solved in time $f(k, n)$, with $n$ being the size of the input and $f$ polynomial in $n$ for fixed $k$.

$$\textsf{fpt} \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq W[\text{SAT}] \subseteq W[P] \subseteq XP.$$
OR
AND
Example transformation
WCS reduced to register allocation

- The tree-width of the control-flow graph is at most one more than the tree-width of the circuit.
- The node where all the live ranges of inputs meet ensures that at most \( r = k \) of the inputs are assigned the value “true”.
- Adjusting the weights we can ensure that finding an optimal assignment of variables to registers results in a “yes” or “no” answer for the WCS problem.
The register allocation problem, when parametrized by the number of registers $r$ is $W[\text{SAT}]$-hard, even for structured programs of tree-width 2.