

The Complexity of Register Allocation

Philipp Klaus Krause

October 29, 2011

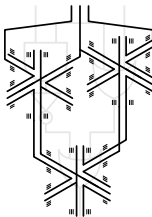


Table of Contents

1 Register Allocation

2 Graph Coloring

3 Hardness

Table of Contents

1 Register Allocation

2 Graph Coloring

3 Hardness

Register Allocation

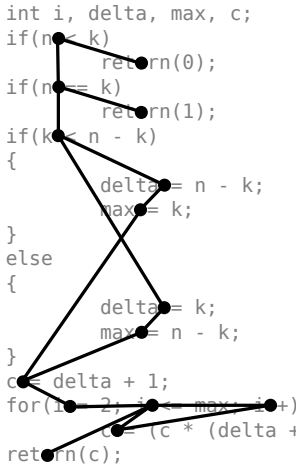
- Programs use variables that have to be stored somewhere
- Compiler decides where to store variables
 - Registers vs. memory
 - Very important for code quality

C Code

```
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);
}
```

Control Flow Graph

```
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);
}
```



The diagram illustrates the control flow graph for the provided C code. It consists of several nodes (black dots) connected by edges (black lines). The nodes are positioned vertically to correspond to the lines of code. The flow starts at the entry node, goes to the first if-statement, then to the second, then to the third. From the third if-statement, the flow branches into two paths based on the condition. Both paths converge at the assignment of 'c'. The flow then enters a loop, which is represented by a horizontal line of nodes connected by arrows pointing right. From the end of the loop, the flow goes to the final return statement.

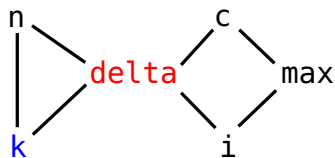
Live Range

```
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);
}
```

The diagram illustrates the live range of the variable `k` in the provided C code. Black dots represent the points where `k` is used or defined. Black lines represent the control flow between these points. A blue line highlights the live range of `k`, starting from its first use in the `if(n < k)` condition, continuing through the `if(n == k)` condition, and then through the `if(k < n - k)` condition. The live range ends at the final use of `k` in the `return(c);` statement. The blue line also encompasses the assignments `delta = n - k;` and `max = k;` within the first `if` block, as well as the assignments `delta = k;` and `max = n - k;` within the `else` block, because these assignments depend on the value of `k`.

Conflict Graph

```
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);
}
```



Register Allocation

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 .

Table of Contents

1 Register Allocation

2 Graph Coloring

3 Hardness

Graph Coloring Register Allocation¹

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.

Tree Width

- 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 bounded².
- For structured programs, and fixed r the decision problem of r -colorability of the conflict graph can be solved in linear time³.
- For structured programs, and fixed r the register allocation problem can be solved in polynomial time⁴.

²M. Thorup, 1998

³H. Bodlaender et alii, 1998

⁴K.

Table of Contents

1 Register Allocation

2 Graph Coloring

3 Hardness

Weighted Circuit Satisfiability

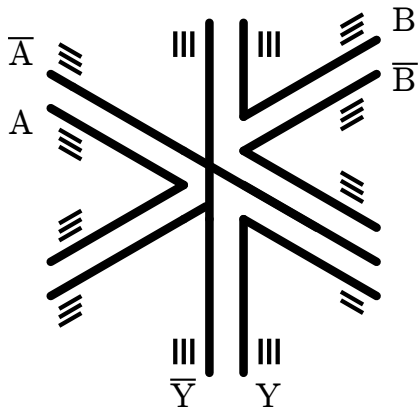
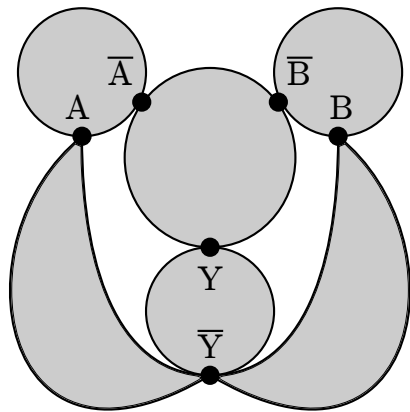
Given a circuit C (input) and an integer k (parameter) the weighted circuit satisfiability problem asks if there is a satisfying assignment for C of weight exactly k (i. e. exactly k of the inputs are set to “true” in the assignment).

Parametrized Complexity Classes

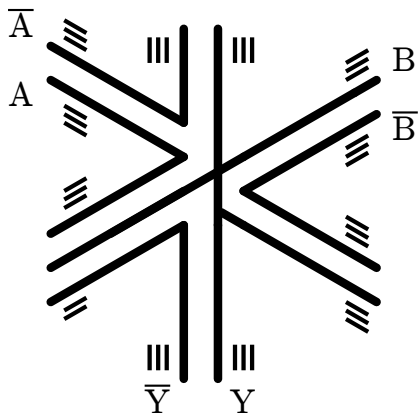
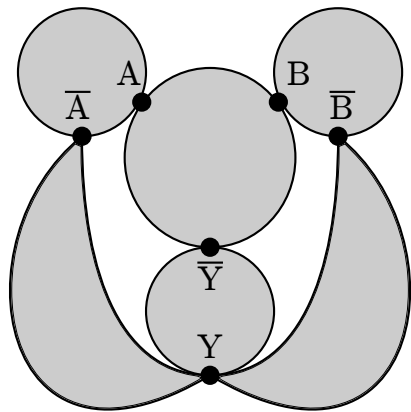
- fpt is the class of all parametrized problems parametrized by k that can be solved in time $f(k)p(n)$ for input size n , computable f and polynomial p .
- $W[t]$ is the class of all parametrized problems that can be fpt -reduced to the WCS problem on weft t , depth d SAT-circuits, for a constant $d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

$$\text{fpt} \subseteq W[1] \subseteq W[2] \subseteq \dots \subseteq W[\text{SAT}] \subseteq W[P] \subseteq XP.$$

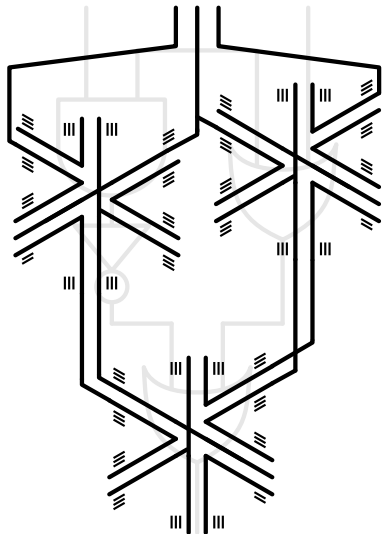
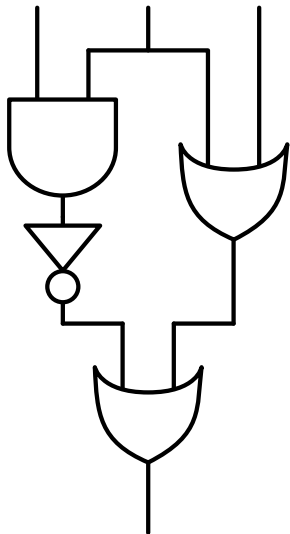
OR



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.

Hardness of register allocation

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.