COM SCI 132 Week 10

Aidan Jan

June 3, 2024

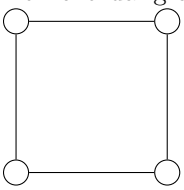
Advanced Register Allocation

Chordal vs. Non-chordal graphs

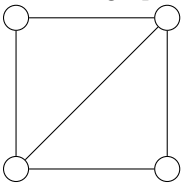
Remember that register allocation can be written as a graph coloring problem where the color represents registers.

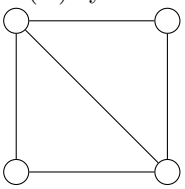
- t

Non-chordal graph:



Chordal graphs: (there are no 4(+)-cycles without chords)





- about 95% of interference graphs contain chords.

Graph Coloring Complexity

Type of Graph	Graph Coloring Complexity
Interval graphs	polynomial time
$\subseteq$ Chordal graphs	polynomial time
$\subseteq$ General graphs	NP-complete

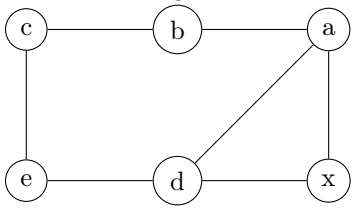
- For chordal and interval graphs, greedy coloring is optimal.
- **Theorem:** A graph G is chorded $\iff$ G has a simplified elimination order.
  - This is good since most register allocation interference graphs are chorded.

Example

Consider the following code:

```
int m(int x, a, d) {     |  x  a  b  c  d  e
    int b, c             |  |  |        |
    if (x > 0) {         |  0  0        |
        e = 0            |              |  *
        c = d            |           *  0  |
    } else {             |     *     0     0
        b = 0            |     |  *
        c = a            |     0  |  *
        e = b            |        0  |     *
    }                    |           |     |
    return e + c         |           0     0
}
```

This code generates an interval graph (not chordal), since (a, b, c, e, d) is a 5-cycle without chords.



Static Single Assignment (SSA)

This form occurs when two different branches in a program assign to the same variable. Here are some examples:

```
if ___ {
    c = 0
} else {
    c = 1
}
__ = c
```

```
c1, c2 = 0
if ___ {
    c1 = 1
} else {
    c2 = 1
}
__ = f(c1, c2)
```

```
c1, c2 = 0
if ___ {
     c1 = 1
     c = c1
} else {
     c2 = 1
     c = c2
}
__ = c
```

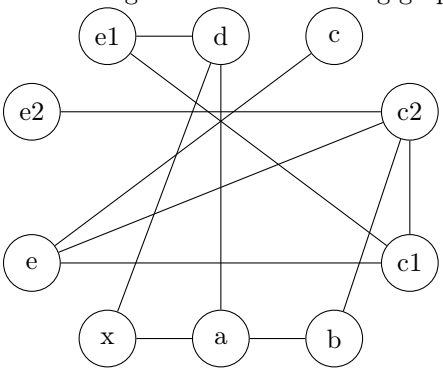
```
Not SSA Form:       SSA Form:

x = new B()         x1 = new B()
x.p(b)              x1.p(b)
...                 ...
x = new C()         x2 = new C()
x.m(C)              x2.m(C)
```

- GCC and LLVM compile to this form.

Example 2

```
int m(int x, a, d) {    |  x  a  b  c1 c2 c  d  e1 e2 e
    int b, c            |  |  |              |
    if (x > 0) {        |  0  0              |
        e1 = 0          |                    |  *
        c1 = d          |           *        0  |
    } else {            |     *     0           0
        b = 0           |     |  *
        c2 = a          |     0  |     *
        e2 = b          |        0     |           *
    }                   |              |           |
    e = f(e1, e2)       |           *  |        0  0  *
    c = f(c1, c2)       |           0  0  *           |
    return e + c        |                 0           0
}
```

This code generates the following graph:



(This is a chordal graph.)

Spilling

- Spilling is NP-complete even for chordal graphs

Compiler:

$$\text{Source} \longrightarrow \text{SSA} \longrightarrow \text{RISC-V}$$
$$\uparrow$$
$$\text{Optimal register allocation in polynomial time}$$

(We can't have optimal register allocation and optimal spilling in polynomial time)

Graphs

$$\text{Interval Graphs} \subseteq \text{Chordal Graphs} \subseteq \text{All undirected graphs}$$

- 100% of procedures **in SSA form** have chordal interference graphs
- 95% of procedures have chorded interference graphs
- All of the procedures are possible as interference graphs

Parallel Assignment

Consider Example 2. Instead of writing

```
e = f(e1, e2)
c = f(c1, c2)
```

write

```
c, e = f(c1, c2), f(e1, e2)
```

This actually *reduces* the complexity because by doing this, `e` and `c1, c2` no longer exist at the same time.