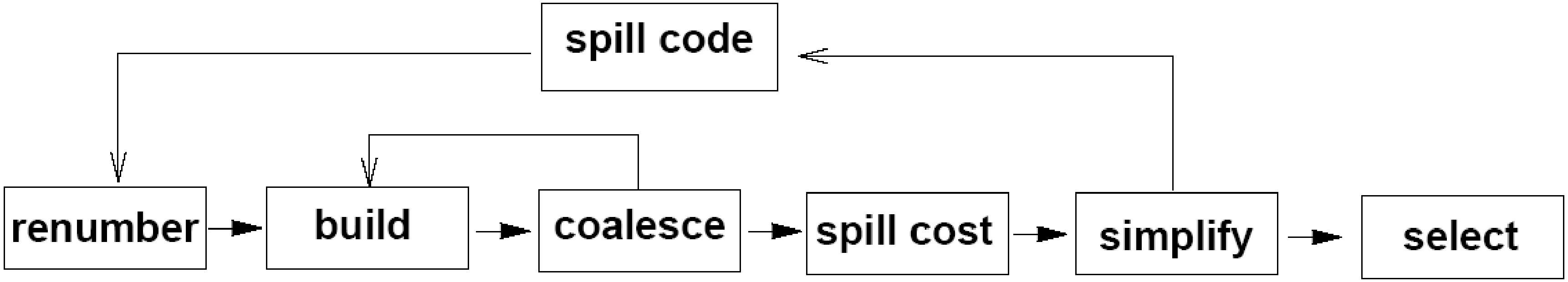
Register Allocation: Some More Advanced Topics

- Optimistic Coloring
- Rematerialization
- Coalescing

Details of Chaitin’s Coloring Phases[82]



spill code

renumber

build

coalesce

spill cost

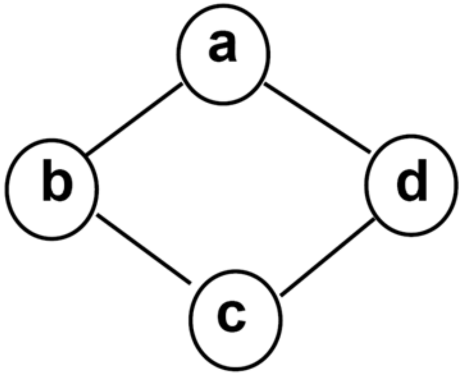
simplify

select

- renumber: rename live ranges
- Build: constructs the interference graph
- Coalesce: merge all pairs of copy-related nodes if they do not interfere
- Spill cost of a node: c x 10^{d}, where c: cost in target machine, d: loop nesting depth
- Simplify: remove and insert to stack or mark spill
- Spill code: if simplify decides to spill any node
- Select: assigns colors to registers

Motivating Problem 1

- For the diamond interference graph that cannot be colored with two registers by Chaitin’s, we can still color it

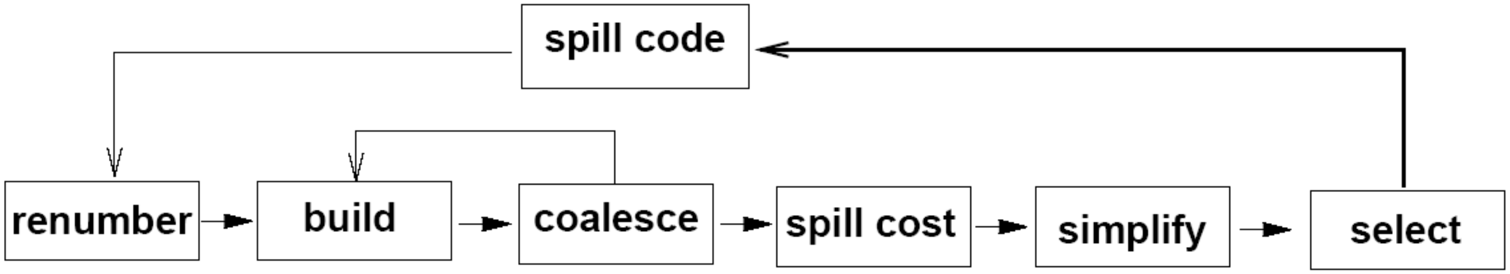


Briggs’ Optimistic Coloring

- Simplify Phase
  - Remove nodes whose degree < n and insert into stack
  - Spill candidates are also inserted in stack for possible coloring
- Select Phase
  - If no color is available for a node, leave the node uncolored and continue with next nodes. Any uncolored node must be one that Chaitin’s would also spill
  - If all nodes receive colors, Done;

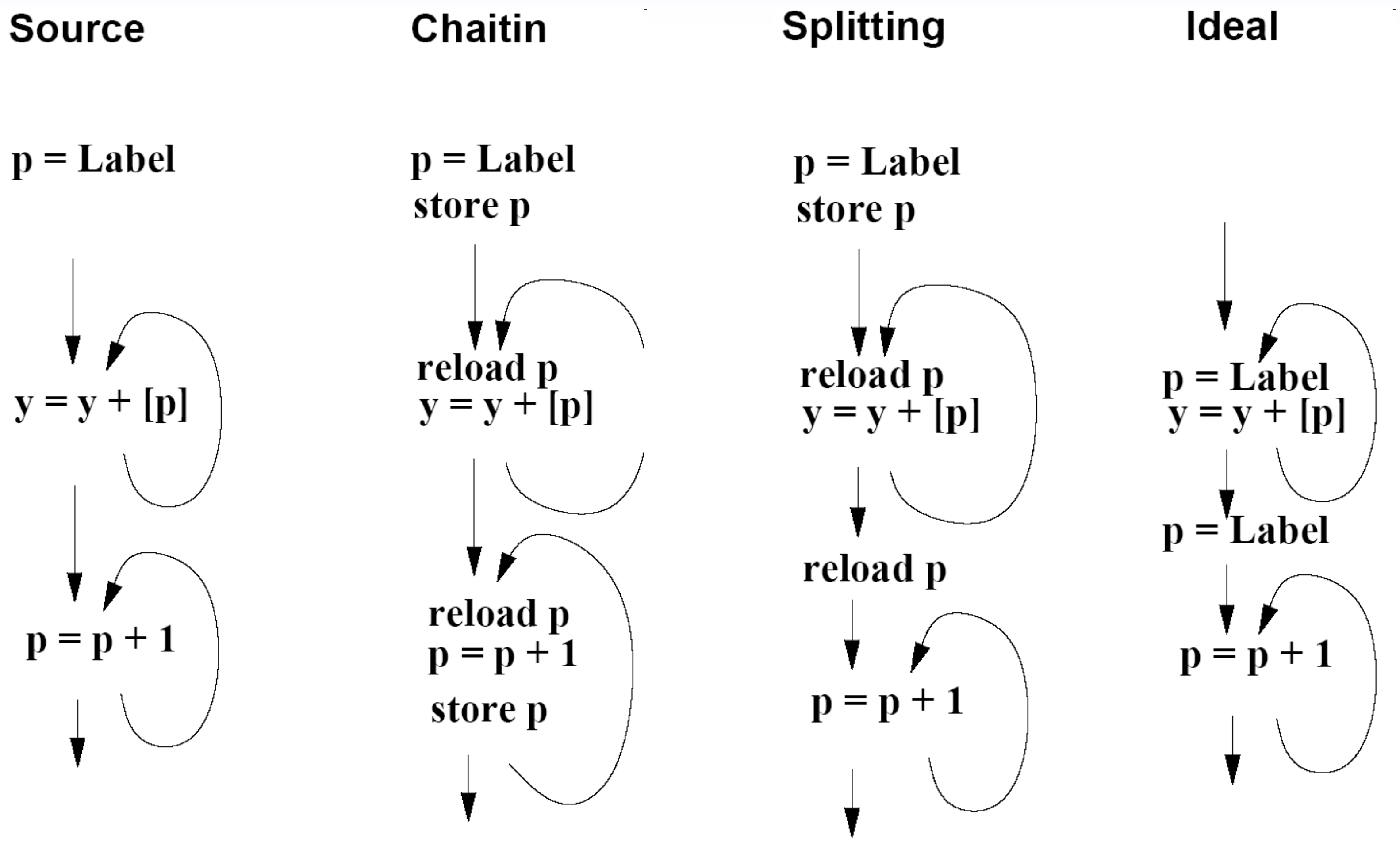
    Otherwise, insert spill code for the corresponding live ranges and repeat from renumber
  - The decision to insert spill code is made at the select phase, not at simplify phase

Details of Briggs' Coloring Phase[89]



- Benefit of Deferring the Spill decision
  - Eliminate some unproductive spills
  - It can color any graph that Chaitin's can, and it can color some graphs that Chaitin's cannot
  - If spilling is necessary, it will spill a subset of live ranges that Chaitin's would spill

Rematerialization [Briggs92]: Idea



Source

```
p = Label
y = y + [p]
p = p + 1
```

Chaitin

```
p = Label
store p
reload p
y = y + [p]
reload p
p = p + 1
store p
```

Splitting

```
p = Label
store p
reload p
y = y + [p]
reload p
p = p + 1
```

Ideal

```
p = Label
y = y + [p]
p = Label
p = p + 1
```

- “Never-Killed” values can be recomputed instead of being spilled and reloaded

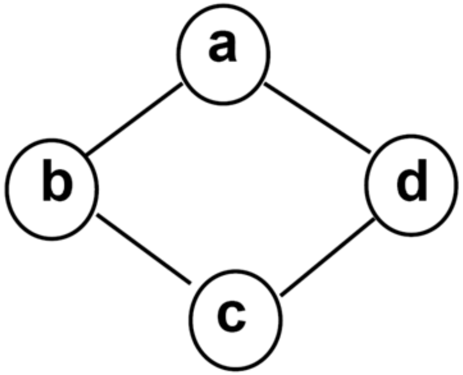
Rematerialization: Opportunities

- Opportunities for Rematerialization
  - Immediate loads (copies) of integers constants
  - Computing a constant offset from the stack pointer or global data area pointer
    - x = sp – 4
    - y = gp + 4

Copy Coalescing

Impact of copy coalescing on graph colorability

- Negative impact
  - Generate significant-degree nodes
- Positive impact
  - Reduce degree of nodes that interferes with both source and target of the copy



Coalescing Approaches

- Aggressive coalescing: Chaitin's coalesces any non-interfering copy-related nodes
  - Best effect on copy elimination & positive impact
  - Suffers seriously from negative impact
- Conservative coalescing: Briggs (and Iterated coalescing) coalesces only when it does not produce a significant-degree node
- Optimistic coalescing: use aggressive coalescing, but if significant-degree nodes are to spilled, split them back to help coloring some splits