Register Allocation

- IR: arbitrary number of variables
- machine: limited number of registers

- Ideal Goal: allocate every IR variable to a register
- Secondary Goal: allocate many IR variables to registers; spill the rest, minimizing spill costs
When can two variables share a register?

- Value of a variable only matters while it is live.
- Two variables can share a register if they are never live at the same time.

**Definition**

A pair of variables *interfere* if there is a program point at which both are live.
Interference Graph

- One vertex for each variable.
- Edge connects two variables if they interfere.

Example [Appel]

live: k, j

g = *(j+12)
h = k - 1
f = g * h
e = *(j+8)
m = *(j+16)
b = *(f)
c = e + 8
d = c
k = m + 4
j = b

live: d, k, j
Register Allocation by Graph Colouring

Goal
Assign a colour (register) to each vertex (variable) so that:
- no two interfering vertices have the same colour, and
- no more than $k$ colours used ($k = \text{number of registers}$)
Goal

Assign a colour (register) to each vertex (variable) so that:
- no two interfering vertices have the same colour, and
- no more than $k$ colours used ($k =$ number of registers)

NP-complete for $k > 2$. 
Heuristic: Simplification

Definition
The **degree** of a vertex \( v \) is the number of edges incident to \( v \).

Theorem
Let \( v \) be a vertex of degree \( < k \) in graph \( G \). Let \( G' \) be the graph obtained by removing \( v \) and all its incident edges from \( G \). If \( G' \) is \( k \)-colourable, then so is \( G \).
The **degree** of a vertex $v$ is the number of edges incident to $v$.

**Theorem**

Let $v$ be a vertex of degree $< k$ in graph $G$. Let $G'$ be the graph obtained by removing $v$ and all its incident edges from $G$. If $G'$ is $k$-colourable, then so is $G$.

**Algorithm**

**Algorithm** $\text{Colour}(G)$:
1: find vertex $v$ of degree $< k$
2: $\text{Colour}(G \setminus v)$
3: assign $v$ a colour distinct from all its neighbours
What if Simplification fails?

**Algorithm**

**Algorithm** $\text{Colour}(G)$:

1. find vertex $v$ of degree $< k$
2. $\text{Colour}(G \setminus v)$
3. assign $v$ a colour distinct from all its neighbours

In step 1, there may not be a vertex of degree $< k$. 
What if Simplification fails?

Algorithm

**Algorithm Colour**$(G)$:
1. find vertex $v$ of degree $< k$
2. Colour$(G \setminus v)$
3. assign $v$ a colour distinct from all its neighbours

In step 1, there may not be a vertex of degree $< k$.

**Option 1 (Chaitin)**

When there is no vertex of degree $< k$, choose a vertex to spill, remove it from the graph, and continue.
What if Simplification fails?

Algorithm

Algorithm \texttt{Colour}(G):

1. find vertex \( v \) of degree \( < k \)
2. \texttt{Colour}(G \setminus v)
3. assign \( v \) a colour distinct from all its neighbours

In step 1, there may not be a vertex of degree \( < k \).

Option 1 (Chaitin)

When there is no vertex of degree \( < k \), choose a vertex to spill, remove it from the graph, and continue.

Option 2 (Briggs)

When there is no vertex of degree \( < k \), just choose a vertex of higher degree.
What if Simplification fails?

Algorithm

Algorithm Colour(G):
1: find vertex ν of degree < k
2: Colour(G \ ν)
3: assign ν a colour distinct from all its neighbours

In step 1, there may not be a vertex of degree < k.

Option 1 (Chaitin)
When there is no vertex of degree < k, choose a vertex to spill, remove it from the graph, and continue.

Option 2 (Briggs)
When there is no vertex of degree < k, just choose a vertex of higher degree. If step 3 fails, spill ν.
Coalescing

Example [Appel]

live: k, j  
g = *(j+12)  
h = k - 1  
f = g * h  
e = *(j+8)  
m = *(j+16)  
b = *(f)  
c = e + 8  
d = c  
k = m + 4  
j = b  
live: d, k, j
Coalescing

Example [Appel]

live: k, j

g = *(j+12)
h = k - 1
f = g * h
e = *(j+8)
m = *(j+16)
b = *(f)
c = e + 8
d = c
k = m + 4
j = b
live: d, k, j
Safe: Coalescing will not change semantics.

It is safe to coalesce $a$ and $b$ if:

- $a$ and $b$ do not interfere, OR
- $a$ and $b$ are never written after the copy
**Safe** Coalescing will not cause additional spills.

**Option 1 [Briggs]**
Coalesce if the coalesced node would have $< k$ neighbours of degree $\geq k$.

**Option 2 [George]**
Coalesce $a$ and $b$ if every node $c$ of degree $\geq k$ interfering with $a$ also interferes with $b$. 