Assignment 3

Yulei Sui

University of Technology Sydney, Australia

Assignment 3: Quiz + A Coding Task

- A quiz (10 points)
 - Data dependence
 - Constraint graph
 - Andersen's points-to analysis

Assignment 3: Quiz + A Coding Task

- A quiz (10 points)
 - Data dependence
 - Constraint graph
 - Andersen's points-to analysis
- One coding task (15 points)
 - Goal: implement Andersen's pointer analysis by solving the constraint graph of a program.
 - Specification and code template:

https://github.com/SVF-tools/SVF-Teaching/wiki/Assignment-3

SVF CPP API

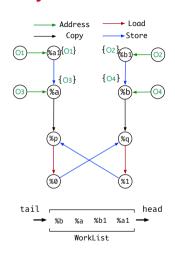
https://github.com/SVF-tools/SVF-Teaching/wiki/SVF-CPP-API

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
                               // O3
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 {
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```

```
Address
 Copy
               ►Store
```

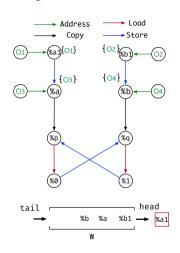
```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        pts(p) = \{o\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ E then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
      foreach p \rightarrow x \in E do
15
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
17
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
                               // O3
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```



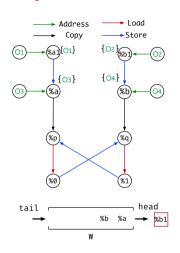
```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        nts(p) = \{0\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ F then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
      foreach p \rightarrow x \in E do
15
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
17
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
                               // O3
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```



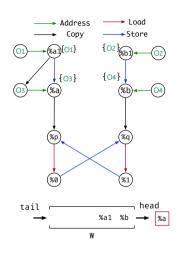
```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        nts(p) = \{0\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ F then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
      foreach p \rightarrow x \in E do
15
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
17
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
                               // O3
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```



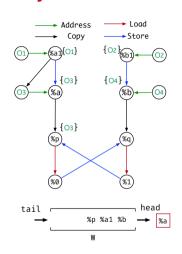
```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        pts(p) = \{o\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ E then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
      foreach p \rightarrow x \in E do
15
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
17
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
                               // O3
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```



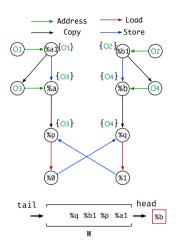
```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        nts(p) = \{0\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ E then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
      foreach p \rightarrow x \in E do
15
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
17
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
                               // O3
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```



```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        nts(p) = \{0\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ F then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
      foreach p \rightarrow x \in E do
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
17
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
                               // O3
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```



```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
  E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        nts(p) = \{0\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
     foreach o E pts(p) do
         foreach store *p = q do// Store rule
            if a→o ∉ F then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
         foreach load r = *p do // load rule
            if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
      foreach p \rightarrow x \in E do
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
17
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```

```
Address
                            → Load
               Copy
                           → Store
                       {O2}(02)
           %a1{O1}
   (01)
{01}
            {03}
                        {04}
             {03}
                        {04}
    tail
                  O3 %a %b1 %p
                     W
```

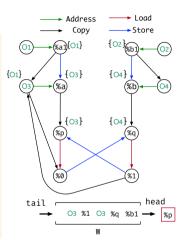
```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        nts(p) = \{0\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ F then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
      foreach p \rightarrow x \in E do
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
                 pushIntoWorklist(x)
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```

```
Address
                            → Load
               Copy
                           → Store
                       {O2}
           (%a){O1}
   (01)
{01}
            {03}
                        {04}
             {03}
                        {04}
           (%0
    tail
                                head
                 %1 O3 %a %b1
                     W
```

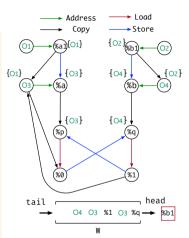
```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        nts(p) = \{0\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ E then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
      foreach p \rightarrow x \in E do
15
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
17
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```



```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        nts(p) = \{0\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ F then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
15
      foreach p \rightarrow x \in E do
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
17
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```



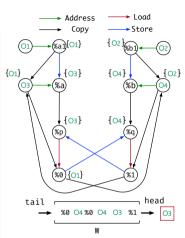
```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        nts(p) = \{0\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ F then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
      foreach p \rightarrow x \in E do
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
                 pushIntoWorklist(x)
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```

```
Address
                              → Load
                 Copy
                             → Store
                         {O2}<sub>(2h1)</sub>-
            (%a){O1}
   (01)
{01}
             {03}
                          {04}
                                        {02}
              {03}
                          {04}
            %р
            %a
    tail
                                    head
            O4 %0 O4 O3 %1 O3
                       W
```

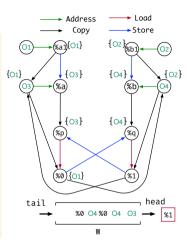
```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        nts(p) = \{0\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ F then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
         foreach load r = *p do // load rule
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
15
      foreach p \rightarrow x \in E do
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
17
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```



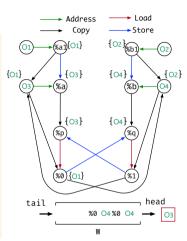
```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        pts(p) = \{o\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ E then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
      foreach p \rightarrow x \in E do
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```



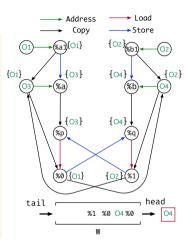
```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        pts(p) = \{o\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ E then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
      foreach p \rightarrow x \in E do
15
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
17
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```



```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        pts(p) = \{o\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ E then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
15
      foreach p \rightarrow x \in E do
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
17
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```



```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        pts(p) = \{o\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ F then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
      foreach p \rightarrow x \in E do
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
                 pushIntoWorklist(x)
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```

```
Address
                               → Load
                 Copy
                              → Store
                          {O2}<sub>(2h1)</sub>-
            (%a){O1}
   (01)
{01}
              {03}
                           {04}
                                        {02,01}
               {03}
                           {04}
             %p
            (%a){\01}
                          {O2 } (%)
     tail
                                     head
                      O4 %1 %0 O4 --
                        W
```

```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        pts(p) = \{o\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ F then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
      foreach p \rightarrow x \in E do
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
17
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```

```
Address
                               → Load
                 Copy
                              → Store
                          {O2}(02)
            (%a){O1}
   (01)
{01}
             {03}
                           {04}
                                        {02,01}
               {03}
                           {04}
             %p
            (%0)<sup>*</sup>{O1}
     tail
                                    head
                    %1 O4 %1 %0
                        W
```

```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        pts(p) = \{o\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ F then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
      foreach p \rightarrow x \in E do
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```

```
Address
                                → Load
                  Copy
                                → Store
                           {O2}(02)
             (%a){O1}
    (01)
{01,02}
               {03}
                            {04}
                                         {02,01}
                {03}
                            {04}
              %p
              (%0)<sup>*</sup>{O1}
      tail
                                     head
                        O3 %1 O4
                         W
```

```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        pts(p) = \{o\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ F then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
15
      foreach p \rightarrow x \in E do
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
                 pushIntoWorklist(x)
18
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```

```
Address
                             → Load
                Copy
                            → Store
                        {O2}
            (%a){O1}
    (01)
{01,02}
             {03}
                         {04}
                                     {02,01}
              {03}
                         {04}
            %р
    {01,621(%0)
     tail
                                 head
                      W
```

```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        nts(p) = \{0\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ F then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
      foreach p \rightarrow x \in E do
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
                 pushIntoWorklist(x)
```

```
define i32 @main() #0 {
entry:
%a1 = alloca i8. alian 1
                               // 01
%b1 = alloca i8, alian 1
                               // O2
%a = alloca i8*, alian 8
                               // 03
%b = alloca i8*, alian 8
                               // 04
store i8* %a1, i8** %a, alian 8
store i8* %b1, i8** %b, alian 8
call void @swap(i8** %a, i8** %b)
ret i32 0
define void @swap(i8** %p, i8** %a)
#0 S
entry:
\%0 = load i8** \%p, alian 8
%1 = load i8** %a, alian 8
store i8* %1, i8** %p, alian 8
store i8* %0. i8** %a. alian 8
ret void
```

```
Address
                              → Load
                 Copy
                             → Store
                         {O2}(02)
            (%a){O1}
    (01)
{01,02}
             {03}
                         {04}
                                     {02,01}
                         {04}
               {03}
             %р
     {01,621(%0)
     tail
                                  head
                   Worklist
```

```
G = < V.E > // Constraint Graph
  V: a set of nodes in graph
   E: a set of edges in graph
  WorkList: a vector of nodes
  foreach address p = &o do // Address rule
        pts(p) = \{o\}
        pushIntoWorklist(p)
  while WorkList ≠ Ø do
      p ← popFromWorklist()
      foreach o E pts(p) do
         foreach store *p = q do// Store rule
             if a→o ∉ F then
                E \leftarrow E \cup \{q \rightarrow o\} // Add copy edge
                pushIntoWorklist(q)
10
         foreach load r = *p do // load rule
11
             if o→r ∉ F then
12
                E \leftarrow E \cup \{o \rightarrow r\} // Add copy edge
13
                pushIntoWorklist(o)
14
15
      foreach p \rightarrow x \in E do
                                    // Copy rule
16
          pts(x) \leftarrow pts(x) \cup pts(p)
          if pts(x) changed then
17
                 pushIntoWorklist(x)
18
```