EXTENDING AlDeSCo BY

Building a data flow graph for java

Lukas Pietzschmann
lukas.pietzschmann@uni-ulm.de

Institute of Software Engineering and
Programming Languages
University of Ulm

10/29/2023

Why even bother?

⊙ **AlDeSCo #83** 😄

Datenfluss wäre hilfreich gewesen

```
if(arr[i] > arr[j])
  // ...
```

What we want to check for

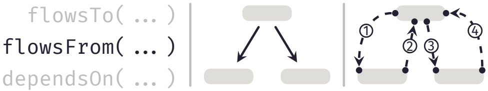
```
boolean needsToSwap = arr[i] > arr[j];
if(needsToSwap)
  // ...
```

What we got

Three steps to success

```java
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

How does `n` propagate through the program?



flowsTo(…)

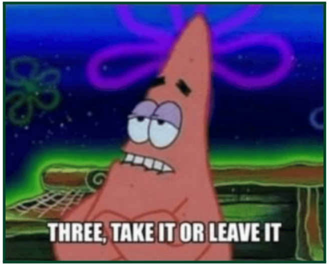
flowsFrom(…)

dependsOn(…)

Select the Operation

```
  flowsTo( ... )
flowsFrom( ... )
dependsOn( ... )
```

We can ask three different questions:

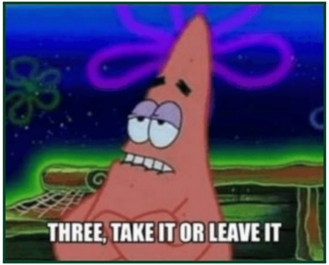


Select the Operation

```
  flowsTo( ... )
flowsFrom( ... )
dependsOn( ... )
```

We can ask three different questions:

- Does `x` **depend on** `y`?

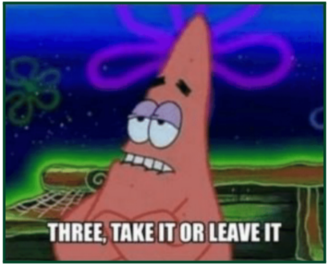


Select the Operation

```
  flowsTo( ... )
flowsFrom( ... )
dependsOn( ... )
```

We can ask three different questions:

- Does x **depend on** y?
- What **flows from** x?

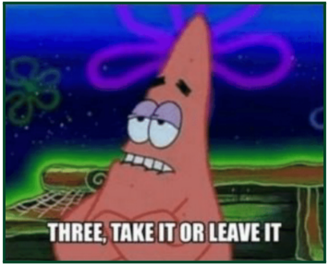


Select the Operation

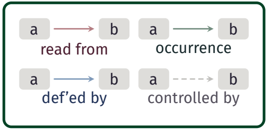
```
  flowsTo( ... )
flowsFrom( ... )
dependsOn( ... )
```

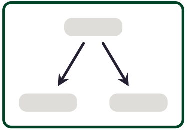
We can ask three different questions:

- Does x **depend on** y?
- What **flows from** x?
- What **flows to** x?



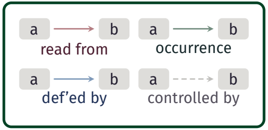
Build the dataflow graph

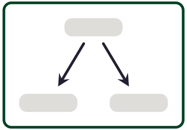




```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

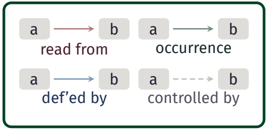
Build the dataflow graph





```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph

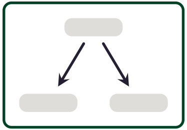


a → b read from

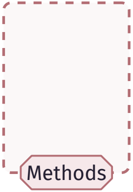
a → b occurrence

a → b def’ed by

a → b controlled by

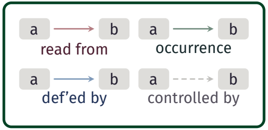


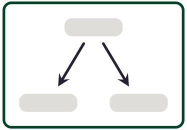
```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

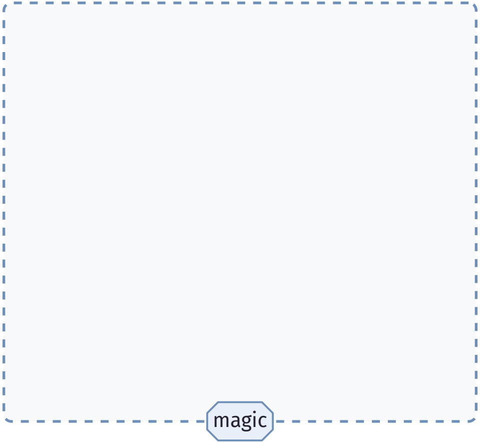


Methods

Build the dataflow graph



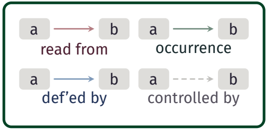


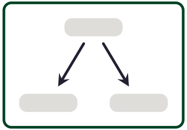


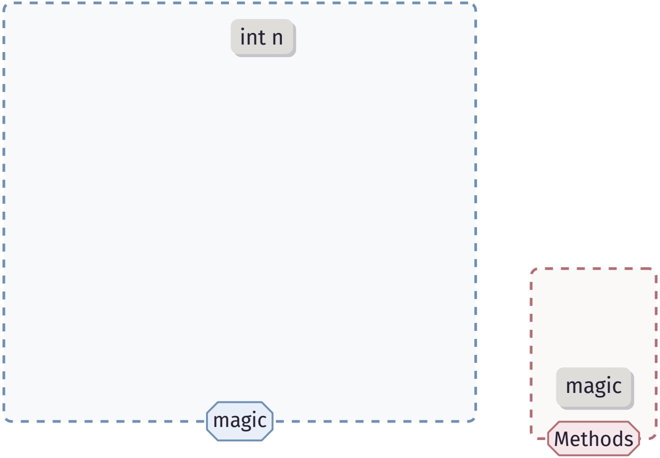
magic
Methods

```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph

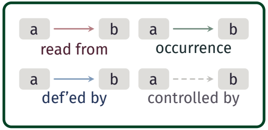


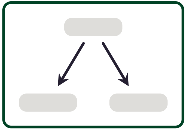


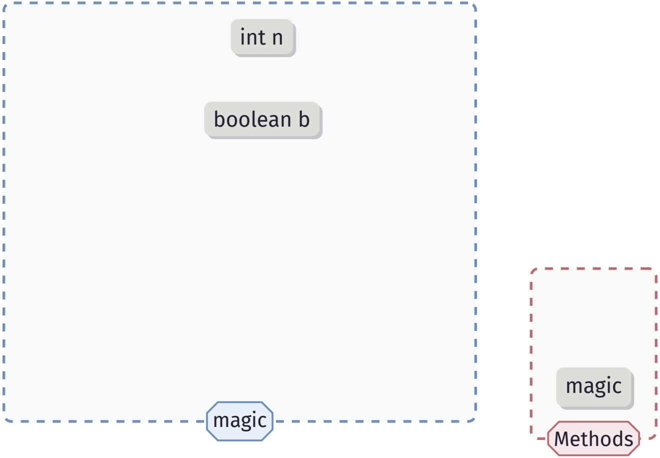


```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph

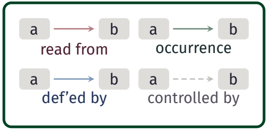


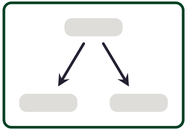


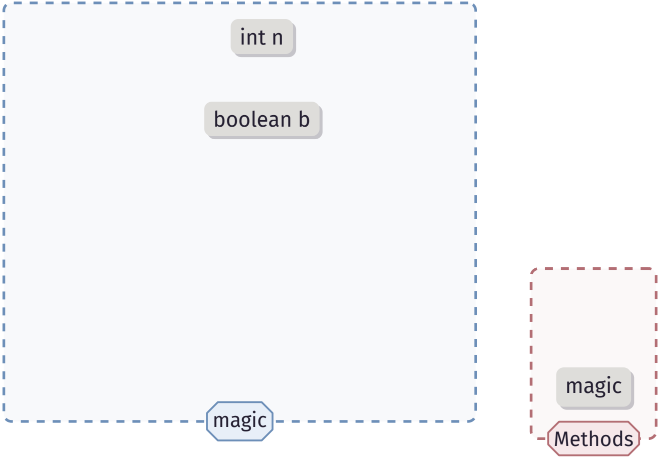


```java
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph

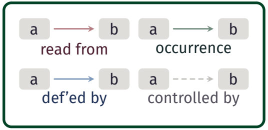


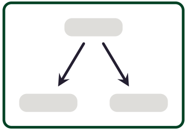


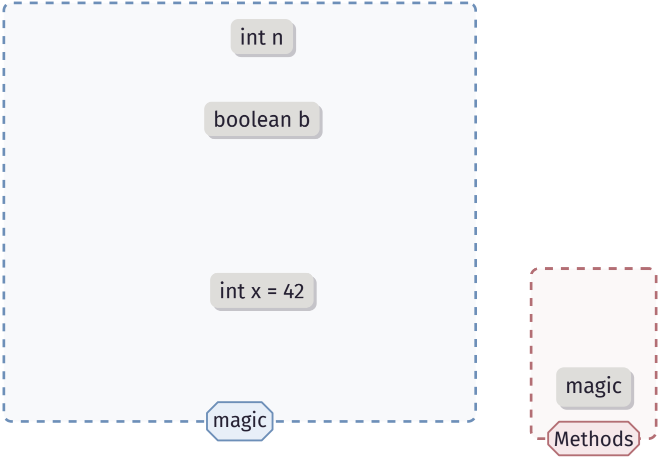


```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph

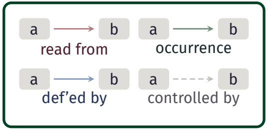


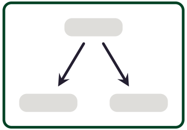


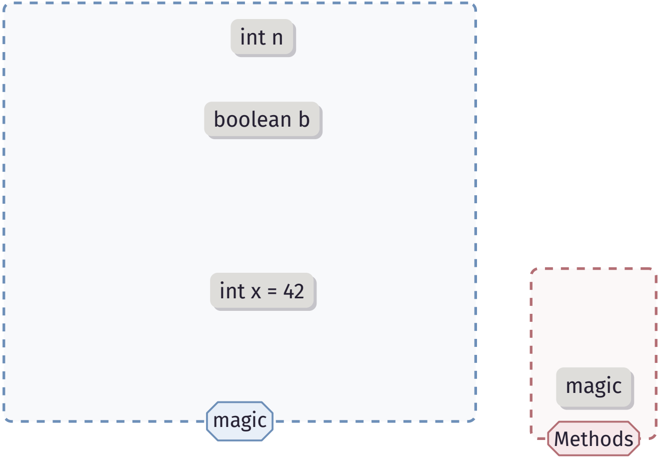


```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph

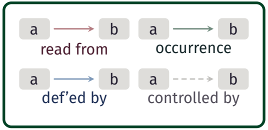


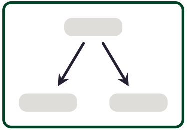


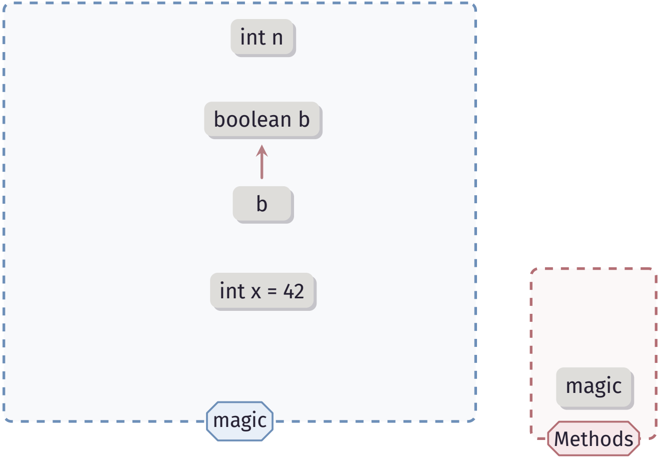


```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph

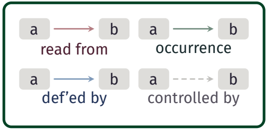


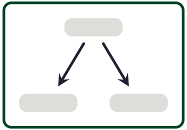


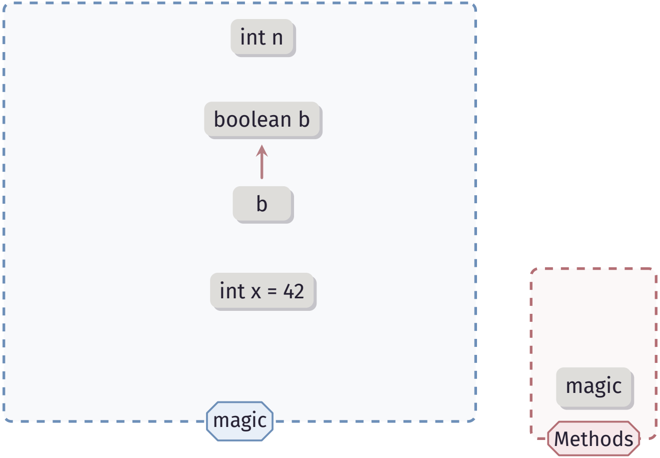


```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph

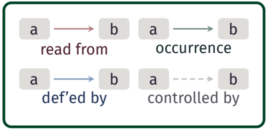


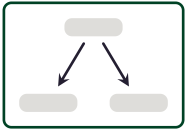


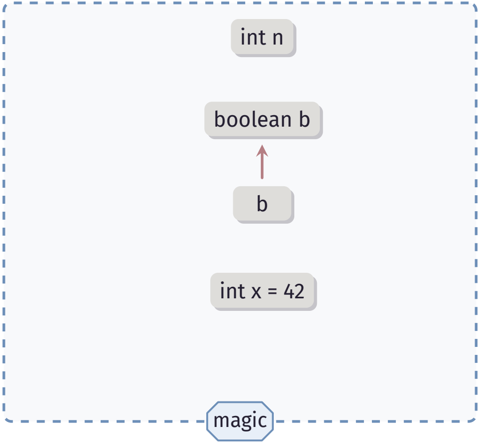


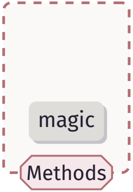
```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph



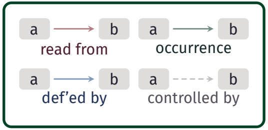


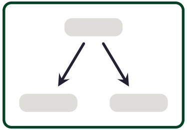


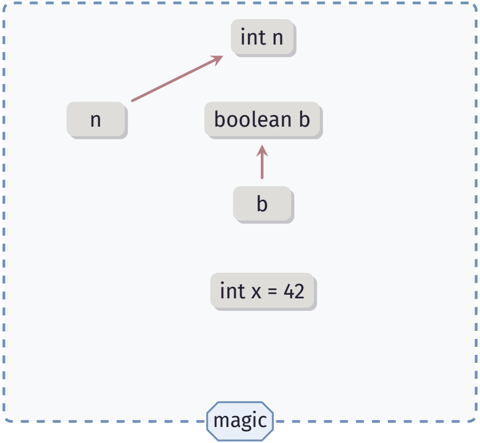


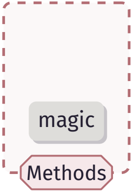
```java
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph



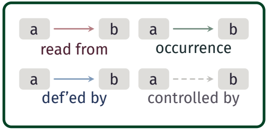


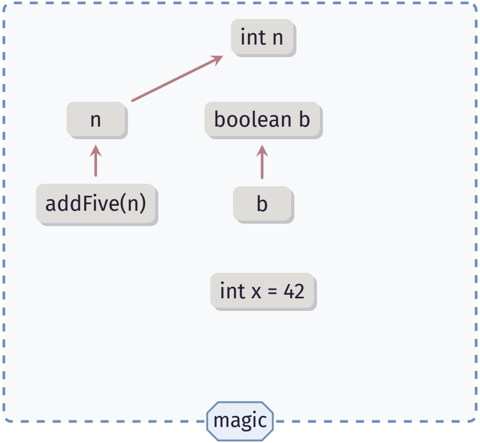


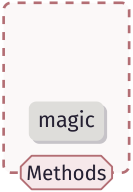


```
class AlDeSCo {
 public int magic(int n, boolean b) {
  int x = 42;
  if(b)
   x = addFive(n);
  else
   x = n;
  return x;
 }
 private int addFive(int m) {
  return m + 5;
 }
}
```

Build the dataflow graph

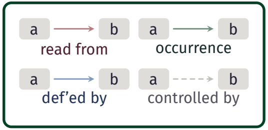


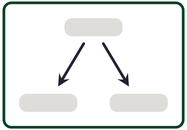


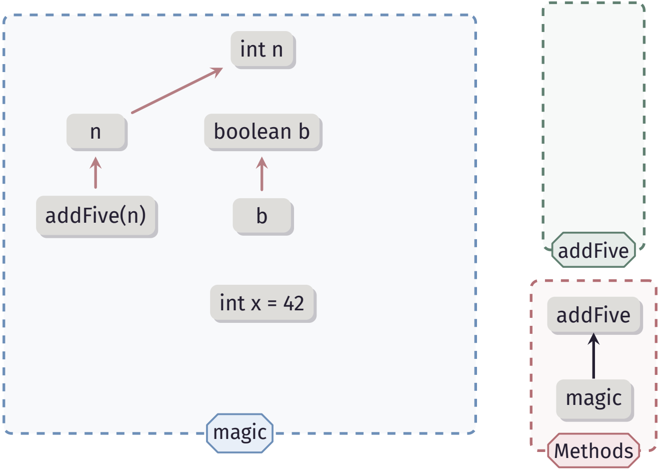


```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph

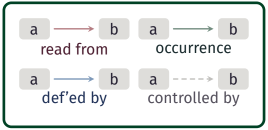


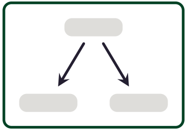


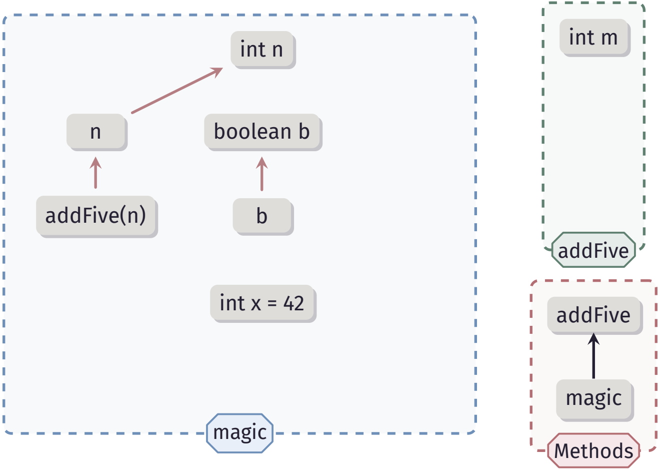


```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph

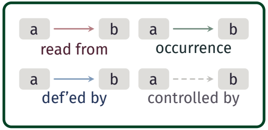


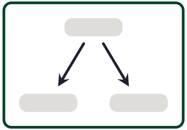


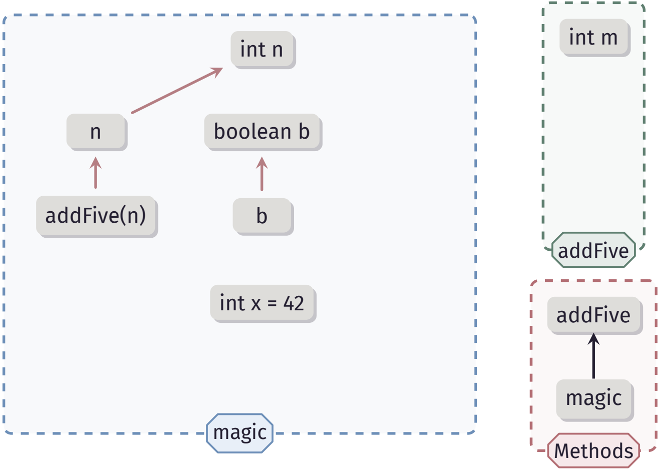


```java
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph

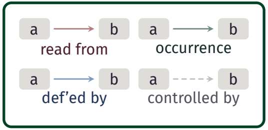


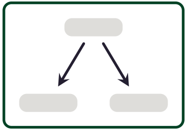


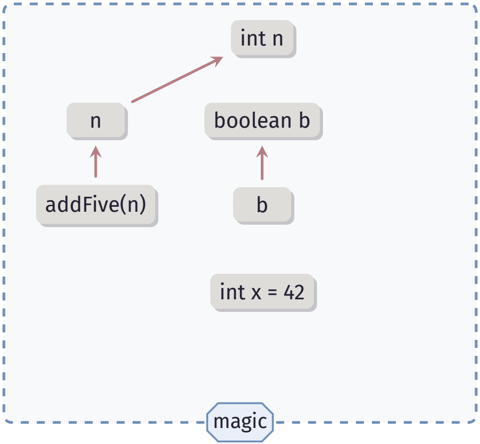


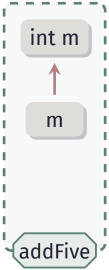
```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

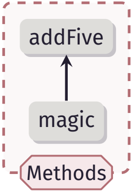
Build the dataflow graph





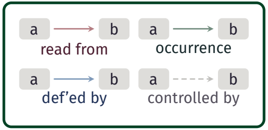


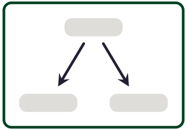


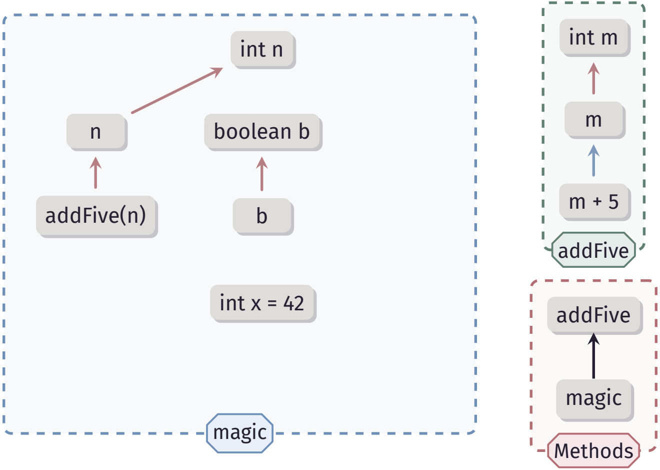


```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph

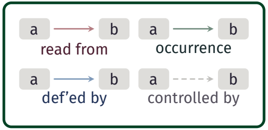


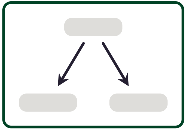


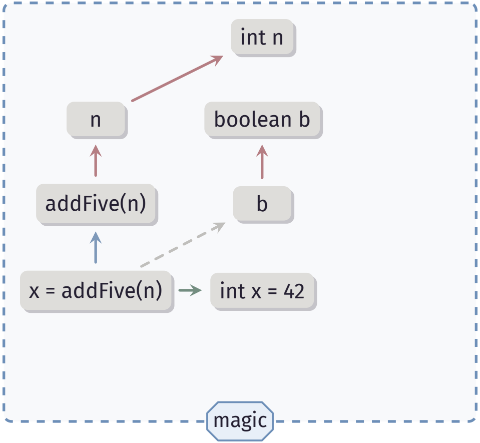


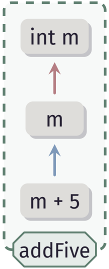
```java
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

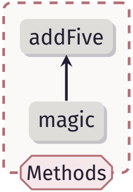
Build the dataflow graph





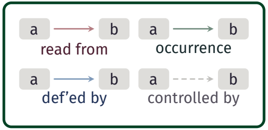


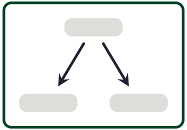


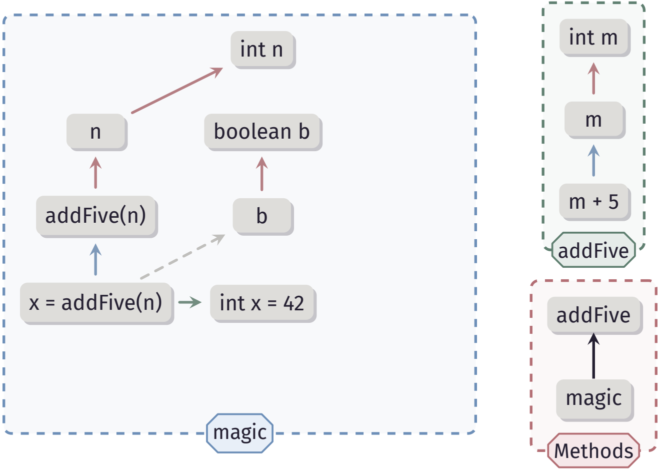


```java
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph

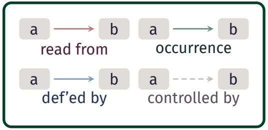


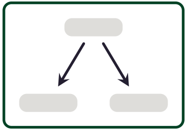


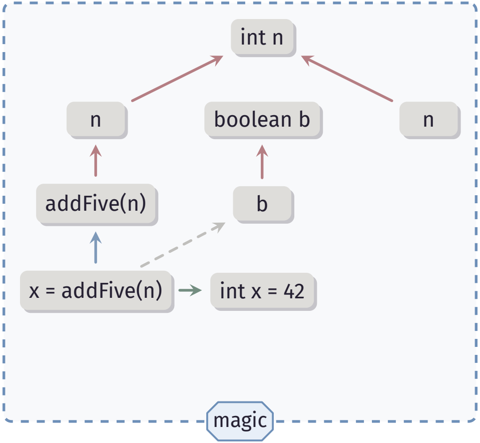


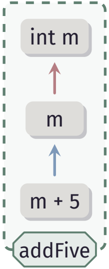
```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

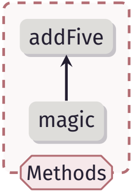
Build the dataflow graph





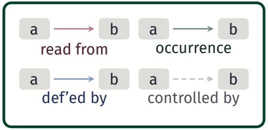


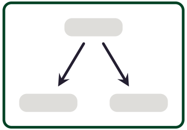


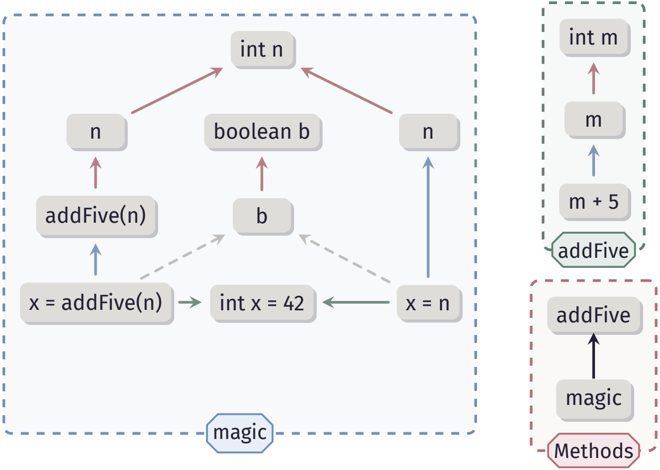


```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph

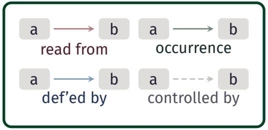


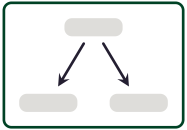


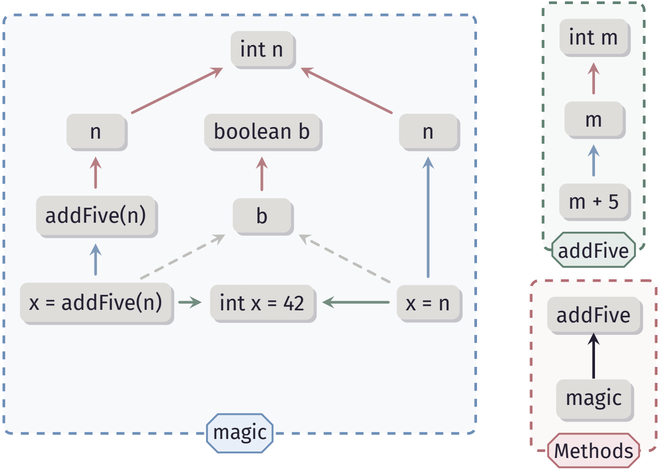


```java
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

Build the dataflow graph

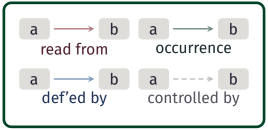


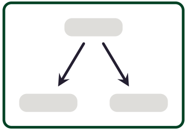


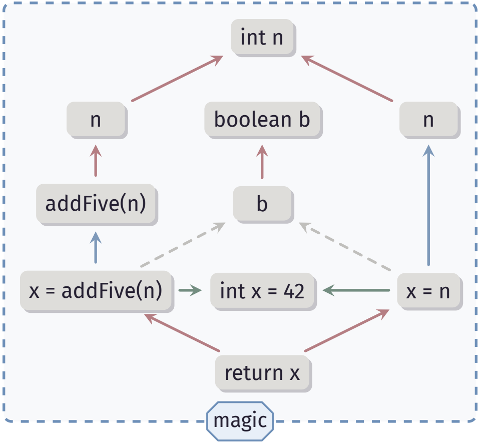


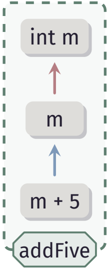
```java
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

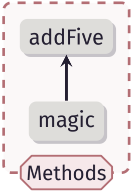
Build the dataflow graph





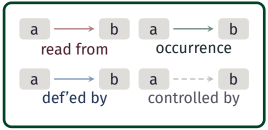


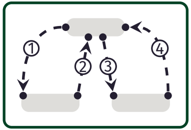




```
class AlDeSCo {
  public int magic(int n, boolean b) {
    int x = 42;
    if(b)
      x = addFive(n);
    else
      x = n;
    return x;
  }
  private int addFive(int m) {
    return m + 5;
  }
}
```

TRAVERSE THE DATAFLOW GRAPH





`flowsTo(a)`

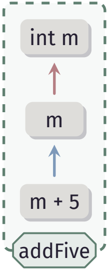
Traverse all paths that lead to a.

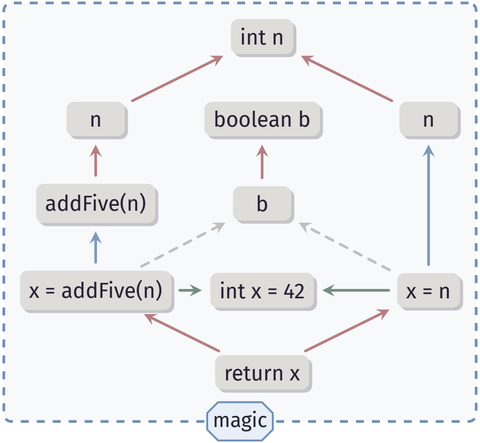
`flowsFrom(a)`

Traverse all paths that start at a.

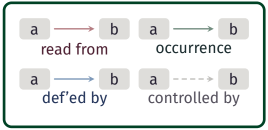
`dependsOn(a, b)`

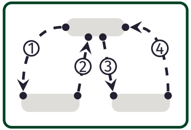
Is there a path between a and b?





Traverse the Dataflow Graph





flowsTo(a)

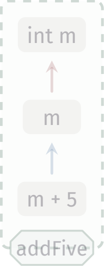
Traverse all paths that lead to a.

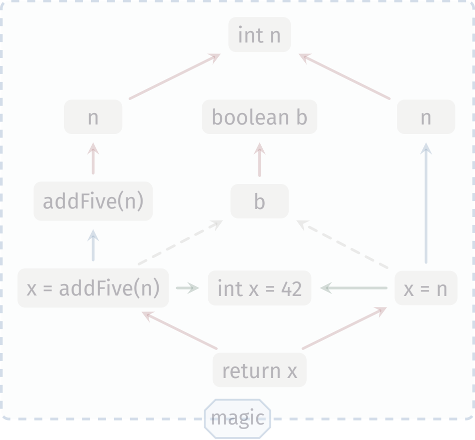
flowsFrom(int n)

Traverse all paths that start at int n.

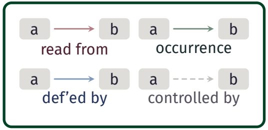
dependsOn(a, b)

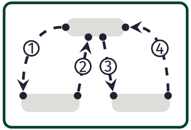
Is there a path between a and b?





Traverse the dataflow graph





`flowsTo(a)`

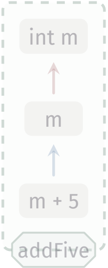
Traverse all paths that lead to a.

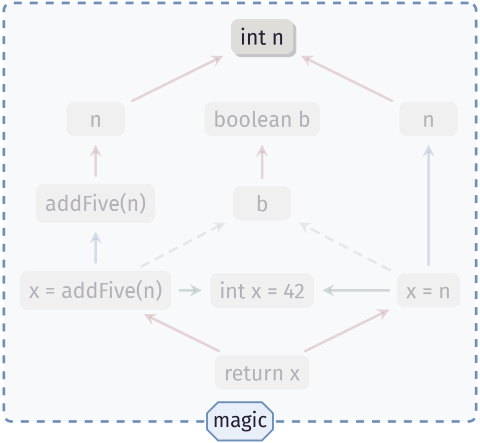
`flowsFrom( int n )`

Traverse all paths that start at int n.

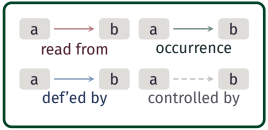
`dependsOn(a, b)`

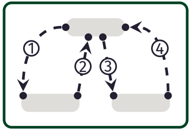
Is there a path between a and b?





Traverse the dataflow graph

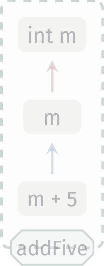


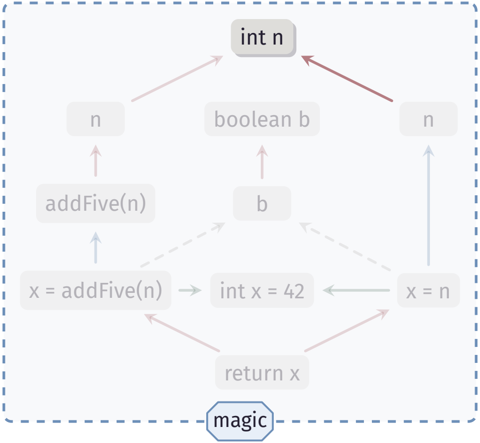


`flowsTo(a)`
: Traverse all paths that lead to a.

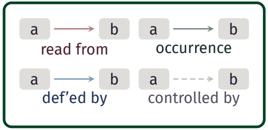
`flowsFrom( int n )`
: Traverse all paths that start at int n.

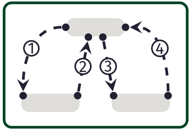
`dependsOn(a, b)`
: Is there a path between a and b?





Traverse the dataflow graph





`flowsTo(a)`

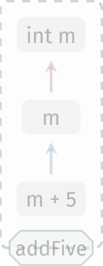
Traverse all paths that lead to a.

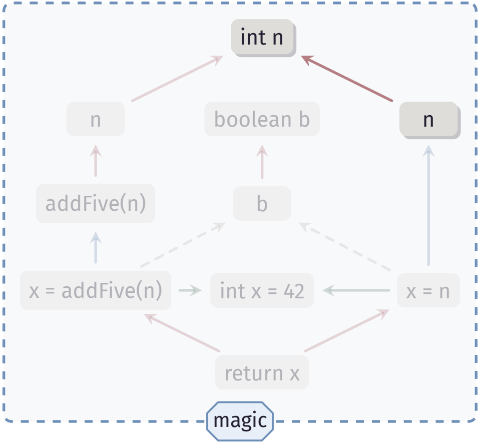
`flowsFrom( int n )`

Traverse all paths that start at int n.

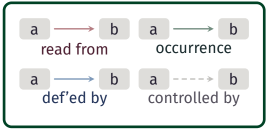
`dependsOn(a, b)`

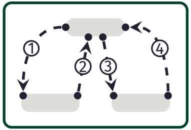
Is there a path between a and b?





Traverse the dataflow graph

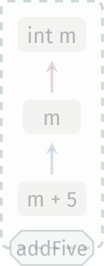


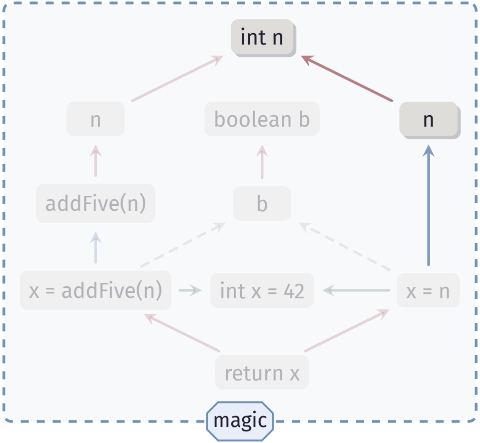


flowsTo(a)
Traverse all paths that lead to a.

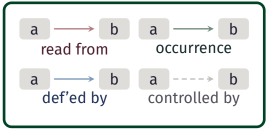
flowsFrom(int n)
Traverse all paths that start at int n.

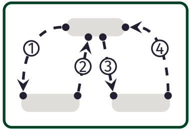
dependsOn(a, b)
Is there a path between a and b?





Traverse the dataflow graph





`flowsTo(a)`

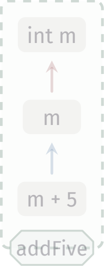
Traverse all paths that lead to a.

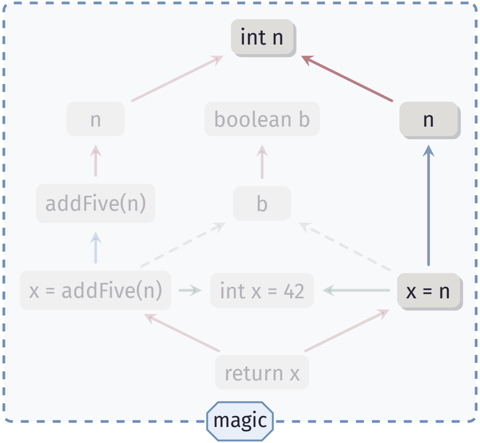
`flowsFrom( int n )`

Traverse all paths that start at int n .

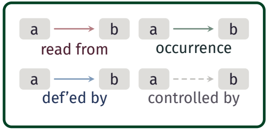
`dependsOn(a, b)`

Is there a path between a and b?





Traverse the Dataflow Graph

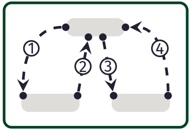


a → b read from

a → b occurrence

a → b def'ed by

a ⇢ b controlled by



`flowsTo(a)`

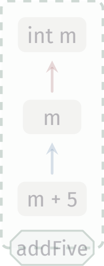
Traverse all paths that lead to a.

`flowsFrom( int n )`

Traverse all paths that start at int n.

`dependsOn(a, b)`

Is there a path between a and b?

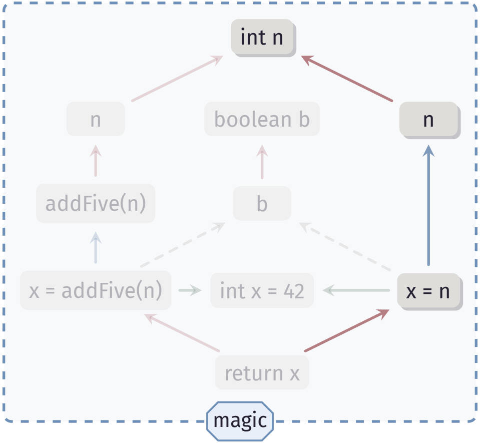


int m

m

m + 5

addFive



int n

n

boolean b

n

addFive(n)

b

x = addFive(n)

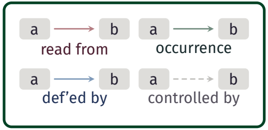
int x = 42

x = n

return x

magic

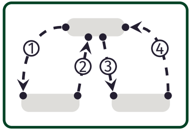
Traverse the Dataflow Graph


a → b read from

a → b occurrence

a → b def'ed by

a → b controlled by



`flowsTo(a)`

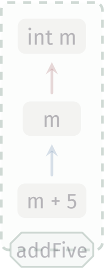
Traverse all paths that lead to a.

`flowsFrom( int n )`

Traverse all paths that start at `int n`.

`dependsOn(a, b)`

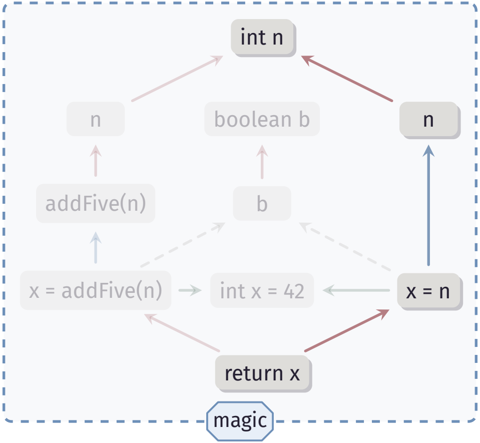
Is there a path between a and b?


int m

m

m + 5

addFive


int n

n

boolean b

n

addFive(n)

b

x = addFive(n)

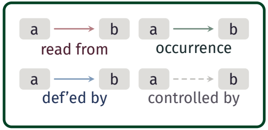
int x = 42

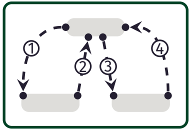
x = n

return x

magic

Traverse the dataflow graph





`flowsTo(a)`

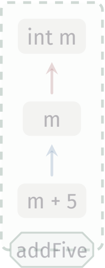
Traverse all paths that lead to a.

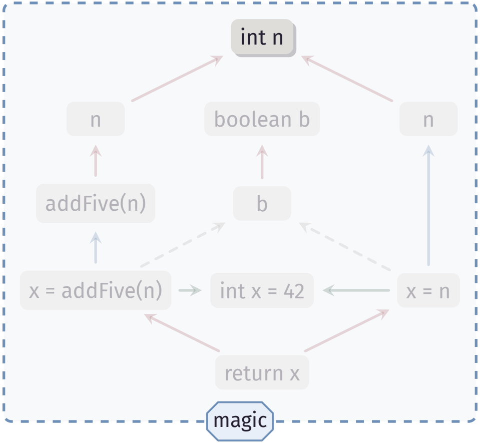
`flowsFrom( int n )`

Traverse all paths that start at int n.

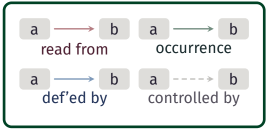
`dependsOn(a, b)`

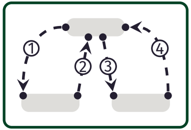
Is there a path between a and b?





TRAVERSE THE DATAFLOW GRAPH





`flowsTo(a)`

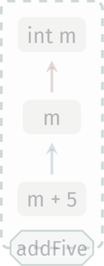
Traverse all paths that lead to a.

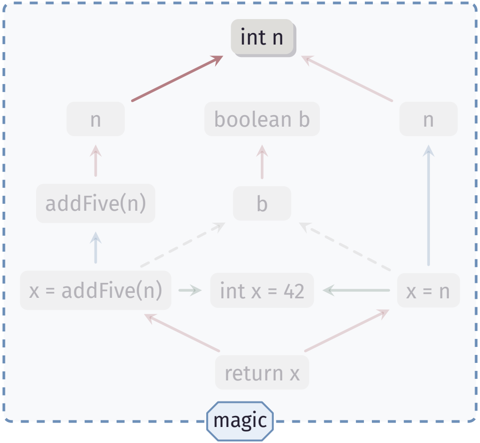
`flowsFrom( int n )`

Traverse all paths that start at int n.

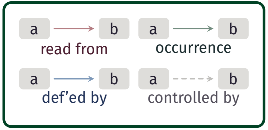
`dependsOn(a, b)`

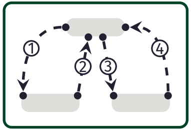
Is there a path between a and b?





Traverse the dataflow graph





flowsTo(a)

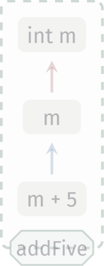
Traverse all paths that lead to a.

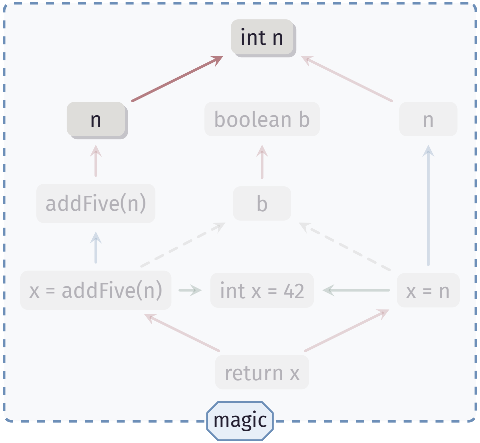
flowsFrom(int n)

Traverse all paths that start at int n.

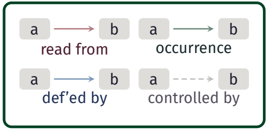
dependsOn(a, b)

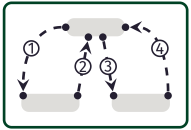
Is there a path between a and b?





Traverse the Dataflow Graph





flowsTo(a)

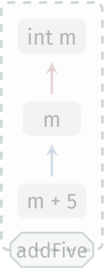
Traverse all paths that lead to a.

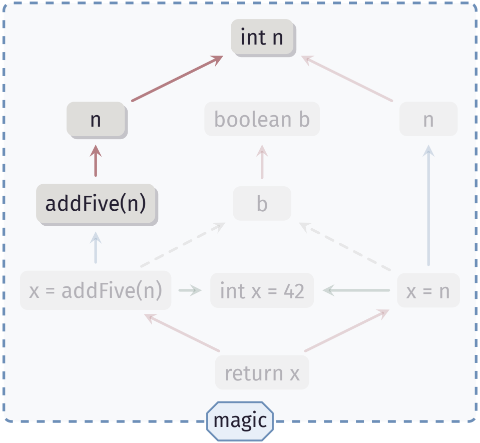
flowsFrom(int n)

Traverse all paths that start at int n.

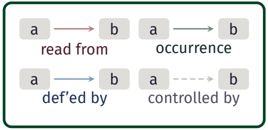
dependsOn(a, b)

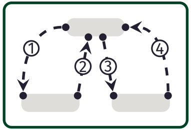
Is there a path between a and b?





TRAVERSE THE DATAFLOW GRAPH





`flowsTo(a)`

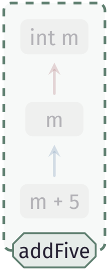
Traverse all paths that lead to a.

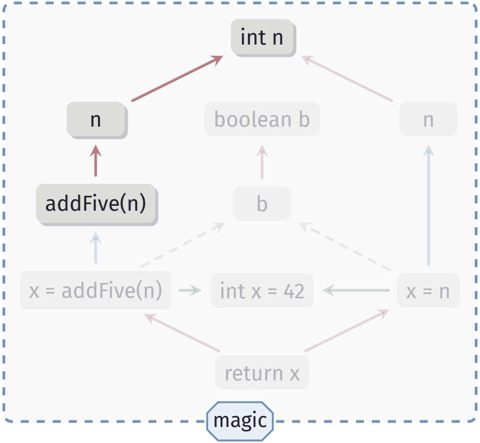
`flowsFrom( int n )`

Traverse all paths that start at `int n`.

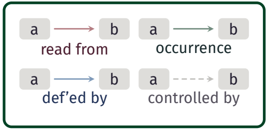
`dependsOn(a, b)`

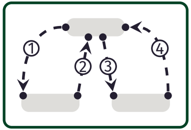
Is there a path between a and b?





Traverse the Dataflow Graph





`flowsTo(a)`

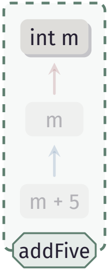
Traverse all paths that lead to a.

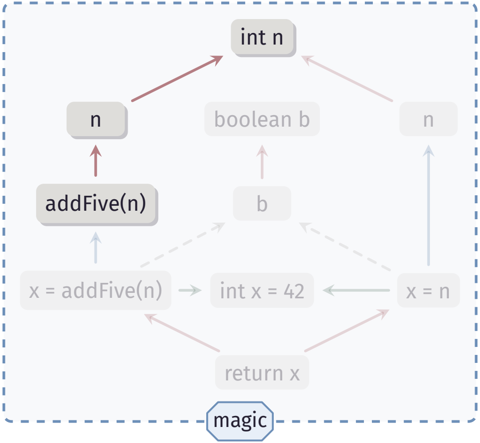
`flowsFrom( int n )`

Traverse all paths that start at int n.

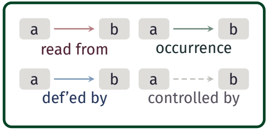
`dependsOn(a, b)`

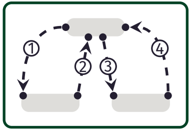
Is there a path between a and b?





Traverse the Dataflow Graph





`flowsTo(a)`

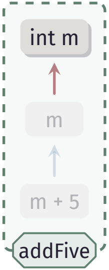
Traverse all paths that lead to a.

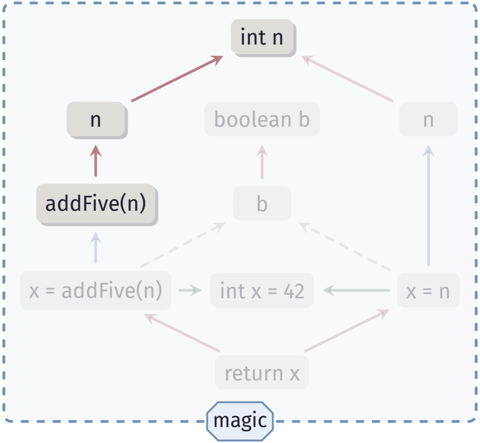
`flowsFrom( int n )`

Traverse all paths that start at int n.

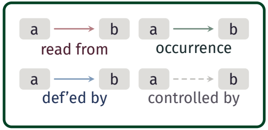
`dependsOn(a, b)`

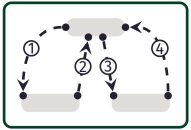
Is there a path between a and b?





Traverse the dataflow graph





`flowsTo(a)`

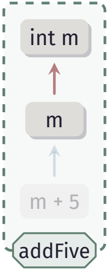
Traverse all paths that lead to a.

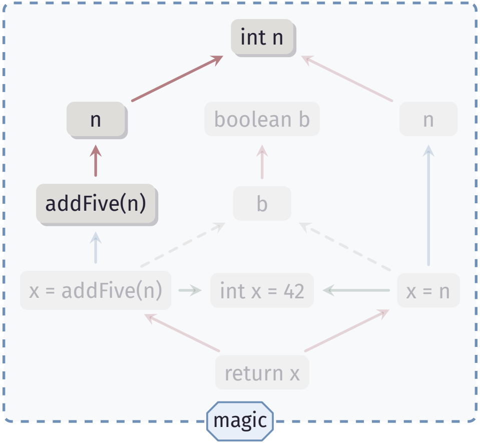
`flowsFrom( int n )`

Traverse all paths that start at int n.

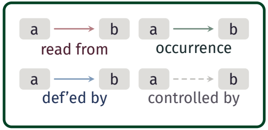
`dependsOn(a, b)`

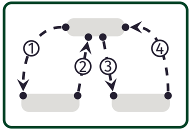
Is there a path between a and b?





Traverse the dataflow graph





`flowsTo(a)`

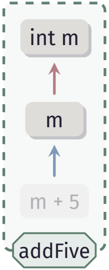
Traverse all paths that lead to a.

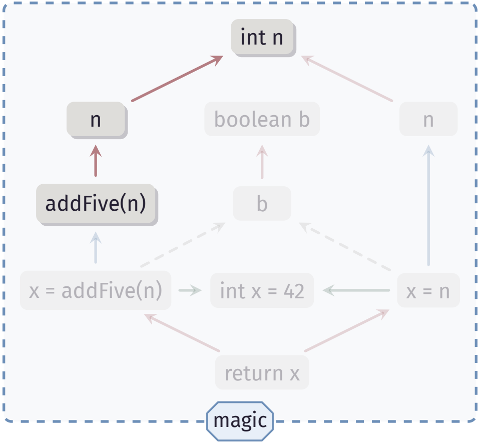
`flowsFrom( int n )`

Traverse all paths that start at int n.

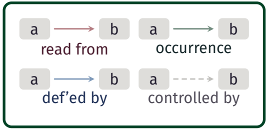
`dependsOn(a, b)`

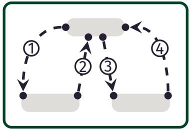
Is there a path between a and b?





Traverse the dataflow graph





flowsTo(a)

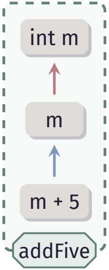
Traverse all paths that lead to a.

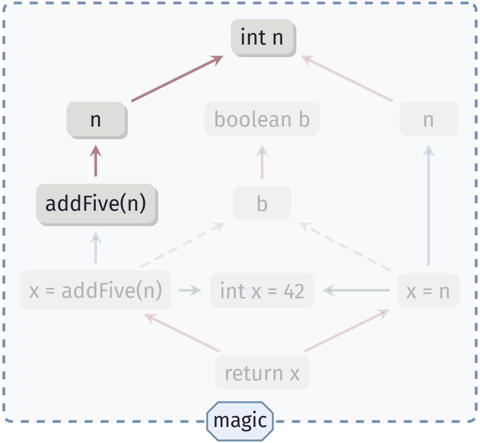
flowsFrom(int n)

Traverse all paths that start at int n .

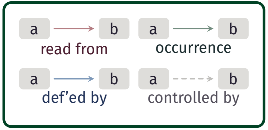
dependsOn(a, b)

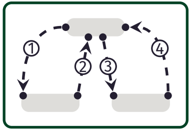
Is there a path between a and b?





Traverse the dataflow graph





flowsTo(a)

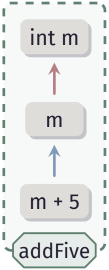
Traverse all paths that lead to a.

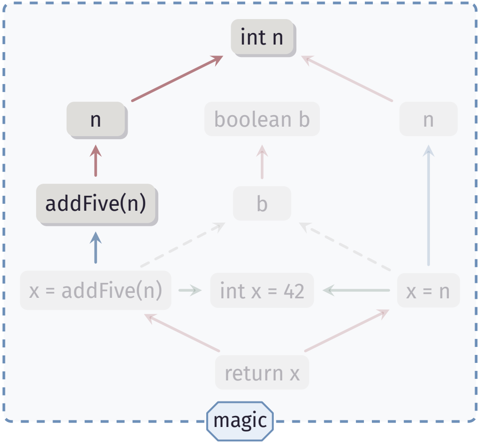
flowsFrom(int n)

Traverse all paths that start at int n.

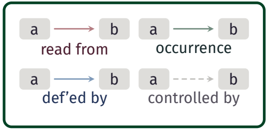
dependsOn(a, b)

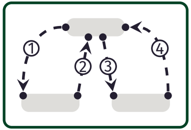
Is there a path between a and b?





Traverse the dataflow graph





`flowsTo(a)`

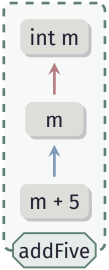
Traverse all paths that lead to a.

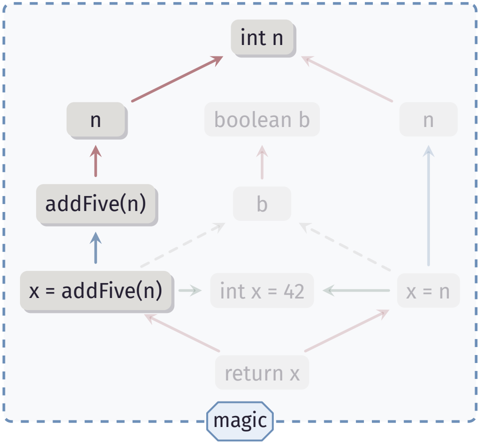
`flowsFrom( int n )`

Traverse all paths that start at `int n`.

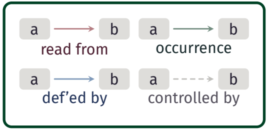
`dependsOn(a, b)`

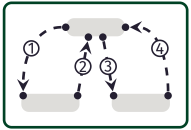
Is there a path between a and b?





Traverse the Dataflow Graph





`flowsTo(a)`

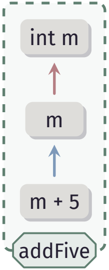
Traverse all paths that lead to a.

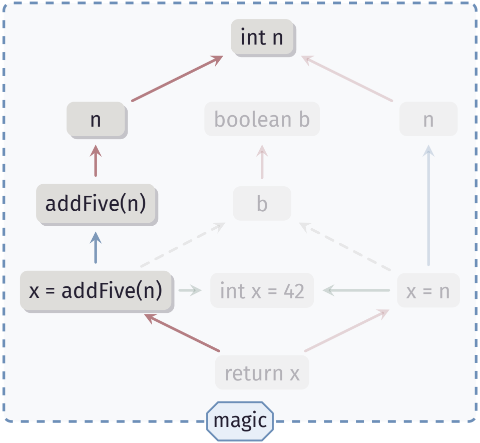
`flowsFrom( int n )`

Traverse all paths that start at int n.

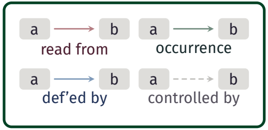
`dependsOn(a, b)`

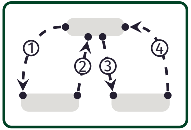
Is there a path between a and b?





Traverse the dataflow graph





`flowsTo(a)`

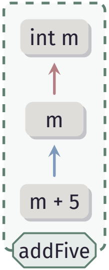
Traverse all paths that lead to a.

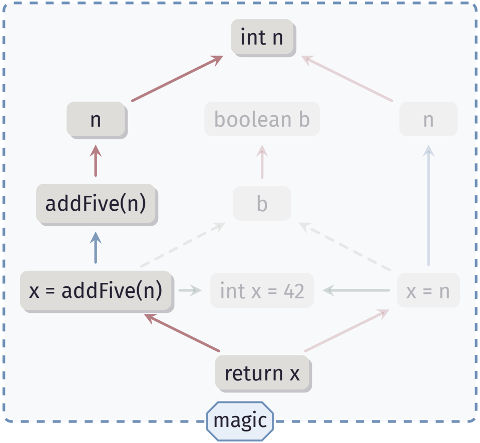
`flowsFrom( int n )`

Traverse all paths that start at int n.

`dependsOn(a, b)`

Is there a path between a and b?





Lukas Pietzschmann

Ulm, 10/29/2023

lukas.pietzschmann@uni-ulm.de