Program Synthesis

Ruben Martins

http://www.cs.cmu.edu/~mheule/15816-f20/
https://cmu.zoom.us/j/93095736668

Automated Reasoning and Satisfiability
October 19, 2020

Carnegie Mellon University

What is Program Synthesis?

Specifications ϕ **Program** P

$$\exists\, P.\ \forall\, x.\ \phi(x, P(x))$$

- Find a program P that for all inputs x meets the specification ϕ

Progress in Program Synthesis

Automating String Processing in Spreadsheets Using Input-Output Examples

Synthesizing Data Structure Transformations from Input-Output Examples *

John K. Feser
Rice University, USA

Component-Based Synthesis for Complex APIs

tins
Austin, USA

Yuepeng Wang
University of Texas at Austin, USA

Program Synthesis from Polymorphic Refinement Types

Scaling Enumerative Program Synthesis via Divide and Conquer *

Rajeev Alur, Arjun Radhakrishna, and Abhishek Udupa

University of Pennsylvania

DEEPCODER: LEARNING TO WRITE PROGRAMS

Matej Balog*
Department of Engineering
University of Cambridge

Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, Daniel Tarlow
Microsoft Research

Program Synthesis is Industry

Tensor reshaping

https://blog.tensorflow.org/2020/08/introducing-tensorflow-coder-tool.html?linkId=98162087

Data manipulation

https://support.microsoft.com/en-us/office/using-flash-fill-in-excel-3f9bcf1e-db93-4890-94a0-1578341f73f7

How do Program Synthesizers Work?

Enumerative Search

Constraint Solving

Stochastic Search

Applications of Program Synthesis

Data Science

Databases

Program Repair

Security

Software Engineering

And many others!

Software Engineering

What happens when you need to implement a new module?

Search → **Copy** → **Paste** → **Edit**

Tedious and error prone approach!

Motivation

Motivation

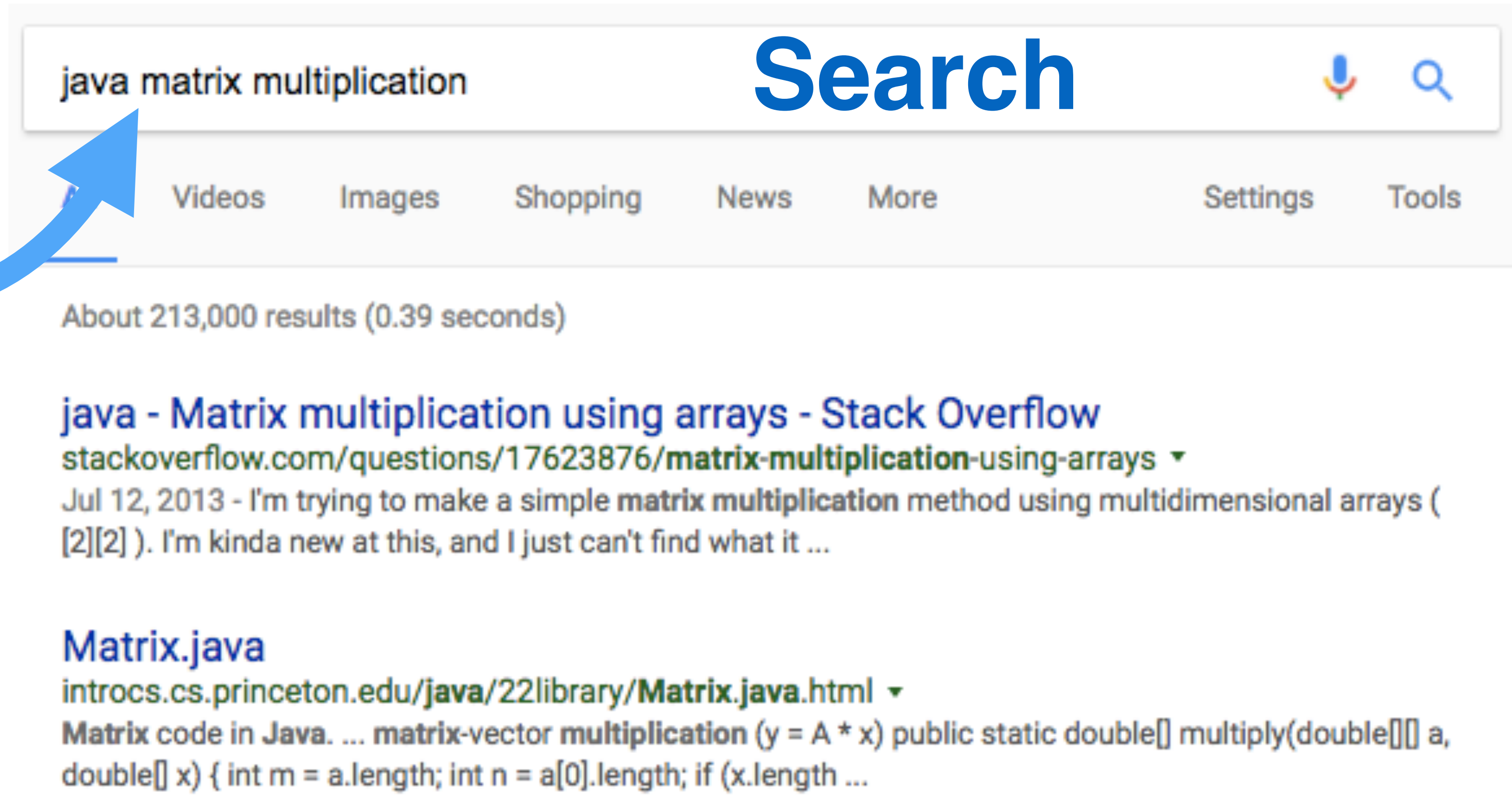
Description: Matrix multiplication

```
void multiply(final Vector<Vector<Double>> first,
              final Vector<Vector<Double>> second,
              Vector<Vector<Double>> res)
```

Motivation

Description: **Matrix multiplication**

```
void multiply(final Vector<Vector<Double>> first,
              final Vector<Vector<Double>> second,
              Vector<Vector<Double>> res)
```



Motivation

Description: **Matrix multiplication**

```
void multiply(final Vector<Vector<Double>> first,
              final Vector<Vector<Double>> second,
              Vector<Vector<Double>> res)
```

Search

java matrix multiplication

Videos Images Shopping News More Settings Tools

About 213,000 results (0.39 seconds)

java - Matrix multiplication using arrays - Stack Overflow
stackoverflow.com/questions/17623876/**matrix-multiplication**-using-arrays
Jul 12, 2013 - I'm trying to make a simple **matrix multiplication** method using multidimensional arrays ([2][2]). I'm kinda new at this, and I just can't find what it ...

Matrix.java
introcs.cs.princeton.edu/**java**/22library/**Matrix.java**.html
Matrix code in **Java**. ... **matrix**-vector **multiplication** (y = A * x) public static double[] multiply(double[][] a, double[] x) { int m = a.length; int n = a[0].length; if (x.length ...

Available Code

Java. Matrix multiplication.

Tested with matrices of different size.

```
public class Matrix {

/**
 * Matrix multiplication method.
 * @param m1 Multiplicand
 * @param m2 Multiplier
 * @return Product
 */
    public static double[][] multiplyByMatrix(double[][] m1, double[][] m2) {
        int m1ColLength = m1[0].length; // m1 columns length
        int m2RowLength = m2.length;    // m2 rows length
        if(m1ColLength != m2RowLength) return null; // matrix multiplication is not possible
        int mRRowLength = m1.length;    // m result rows length
        int mRColLength = m2[0].length; // m result columns length
        double[][] mResult = new double[mRRowLength][mRColLength];
        for(int i = 0; i < mRRowLength; i++) {         // rows from m1
            for(int j = 0; j < mRColLength; j++) {     // columns from m2
                for(int k = 0; k < m1ColLength; k++) { // columns from m1
                    mResult[i][j] += m1[i][k] * m2[k][j];
                }
            }
        }
        return mResult;
    }
```

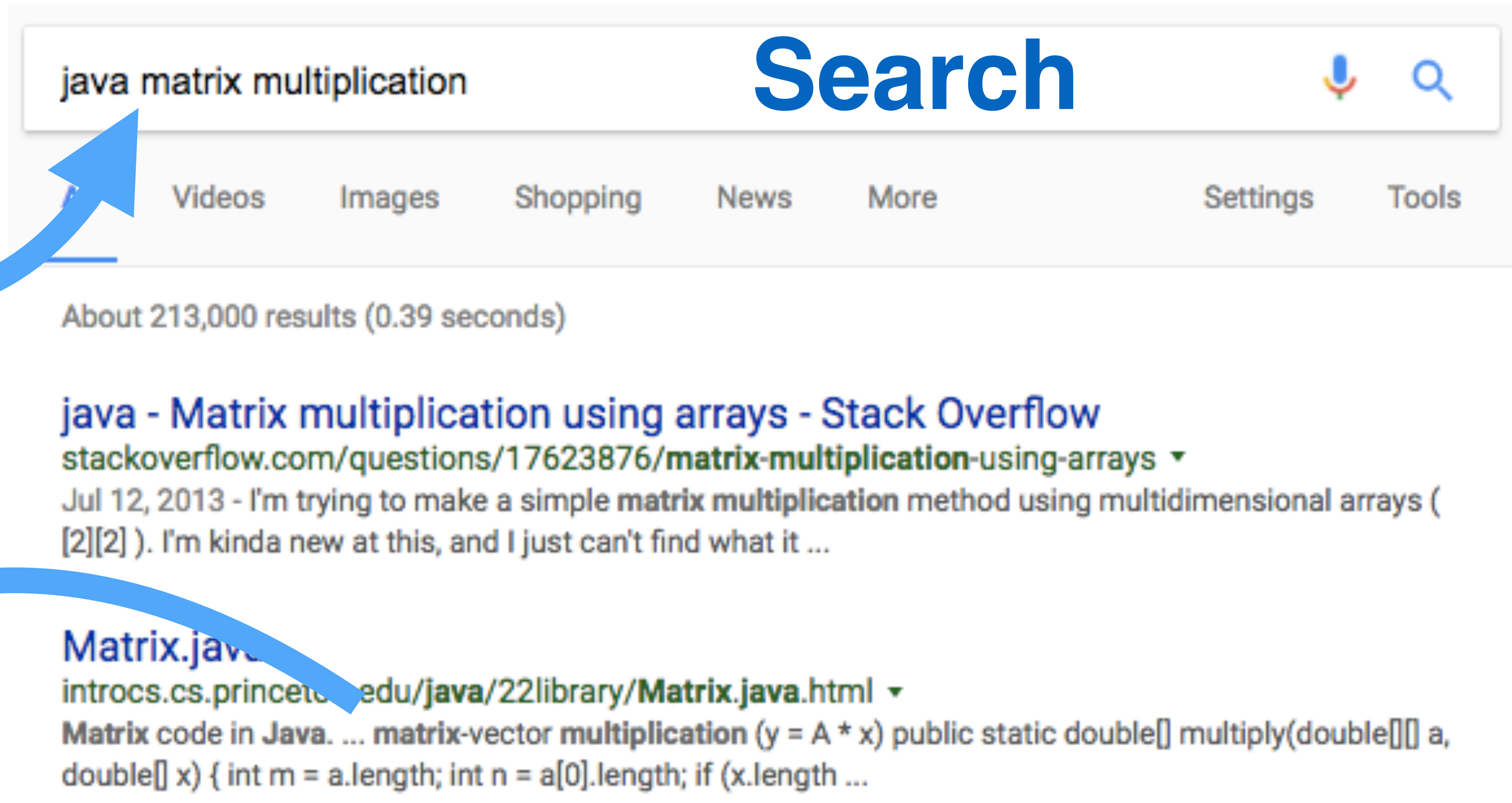
```
double[][] multiplyByMatrix(
      double[][] m1, double[][] m2)
```

Motivation

Description: Matrix multiplication

```
void multiply(final Vector<Vector<Double>> first,
              final Vector<Vector<Double>> second,
              Vector<Vector<Double>> res)
```

Search



Available Code

Java. Matrix multiplication.

Tested with matrices of different size.

```
public class Matrix {

/**
 * Matrix multiplication method.
 * @param m1 Multiplicand
 * @param m2 Multiplier
 * @return Product
 */
    public static double[][] multiplyByMatrix(double[][] m1, double[][] m2) {
        int m1ColLength = m1[0].length; // m1 columns length
        int m2RowLength = m2.length;    // m2 rows length
        if(m1ColLength != m2RowLength) return null; // matrix multiplication is not possible
        int mRRowLength = m1.length;    // m result rows length
        int mRColLength = m2[0].length; // m result columns length
        double[][] mResult = new double[mRRowLength][mRColLength];
        for(int i = 0; i < mRRowLength; i++) {         // rows from m1
            for(int j = 0; j < mRColLength; j++) {     // columns from m2
                for(int k = 0; k < m1ColLength; k++) { // columns from m1
                    mResult[i][j] += m1[i][k] * m2[k][j];
                }
            }
        }
        return mResult;
    }
```

```
double[][] multiplyByMatrix(
      double[][] m1, double[][] m2)
```

Edit

Desired Code

Motivation

Description: **Matrix multiplication**

```
void multiply(final Vector<Vector<Double>> first,
              final Vector<Vector<Double>> second,
              Vector<Vector<Double>> res)
```

Search

java matrix multiplication

Videos Images Shopping News More Settings Tools

About 213,000 results (0.39 seconds)

java - Matrix multiplication using arrays - Stack Overflow
stackoverflow.com/questions/17623876/**matrix-multiplication**-using-arrays
Jul 12, 2013 - I'm trying to make a simple **matrix multiplication** method using multidimensional arrays ([2][2]). I'm kinda new at this, and I just can't find what it ...

Matrix.java
introcs.cs.princeton.edu/**java**/22library/**Matrix.java**.html
Matrix code in **Java**. ... **matrix**-vector **multiplication** (y = A * x) public static double[] multiply(double[][] a, double[] x) { int m = a.length; int n = a[0].length; if (x.length ...

Available Code

Java. Matrix multiplication.

Tested with matrices of different size.

```
public class Matrix {

/**
 * Matrix multiplication method.
 * @param m1 Multiplicand
 * @param m2 Multiplier
 * @return Product
 */
    public static double[][] multiplyByMatrix(double[][] m1, double[][] m2) {
        int m1ColLength = m1[0].length; // m1 columns length
        int m2RowLength = m2.length;    // m2 rows length
        if(m1ColLength != m2RowLength) return null; // matrix multiplication is not possible
        int mRRowLength = m1.length;    // m result rows length
        int mRColLength = m2[0].length; // m result columns length
        double[][] mResult = new double[mRRowLength][mRColLength];
        for(int i = 0; i < mRRowLength; i++) {         // rows from m1
            for(int j = 0; j < mRColLength; j++) {     // columns from m2
                for(int k = 0; k < m1ColLength; k++) { // columns from m1
                    mResult[i][j] += m1[i][k] * m2[k][j];
                }
            }
        }
        return mResult;
    }
```

```
double[][] multiplyByMatrix(
    double[][] m1, double[][] m2)
```

Edit

Desired Code

Long time spent on search
Significant effort required to edit!

Traditional Approach

Available Code ≠ Desired Code

Significant effort required to edit!

Traditional Approach

Available Code + **Improper Edit** = **Buggy Code**

Bugs are easily integrated while editing!

Program Synthesis

- What if the program could write itself?

Description +

= Desired Code

Goal

Description

Desired Code

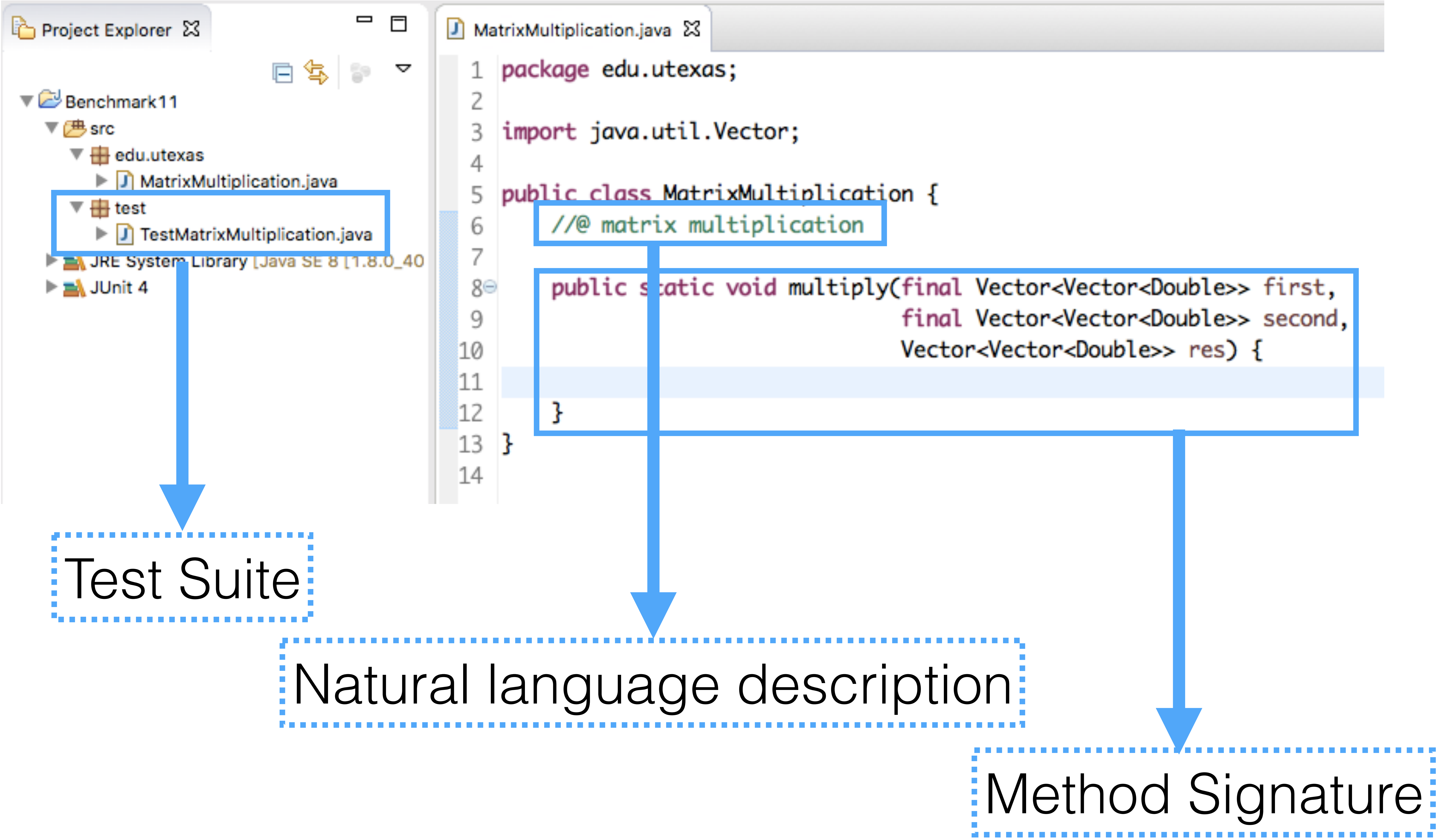
Goal

Description + Click! = **Desired Code**

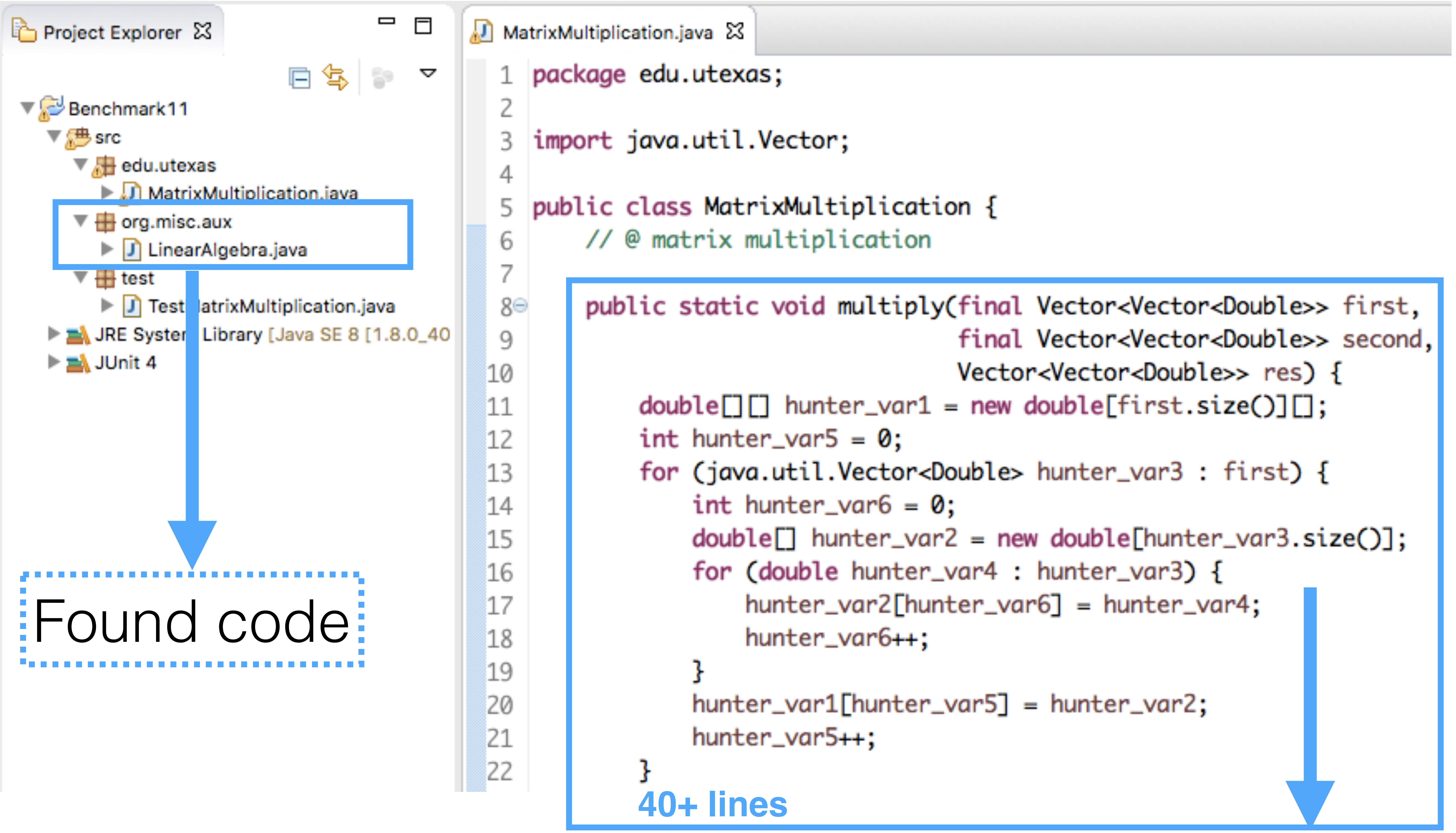
Goal

- Search and reuse code **automatically**
- Method description
  1. Natural language description
  2. Method signature
  3. Test suite

Hunter


Project Explorer
Benchmark11
src
edu.utexas
MatrixMultiplication.java
test
TestMatrixMultiplication.java
JRE System Library [Java SE 8 [1.8.0_40
JUnit 4
MatrixMultiplication.java
package edu.utexas;
import java.util.Vector;
public class MatrixMultiplication {
//@ matrix multiplication
public static void multiply(final Vector<Vector<Double>> first,
final Vector<Vector<Double>> second,
Vector<Vector<Double>> res) {
}
}
Test Suite
Natural language description
Method Signature

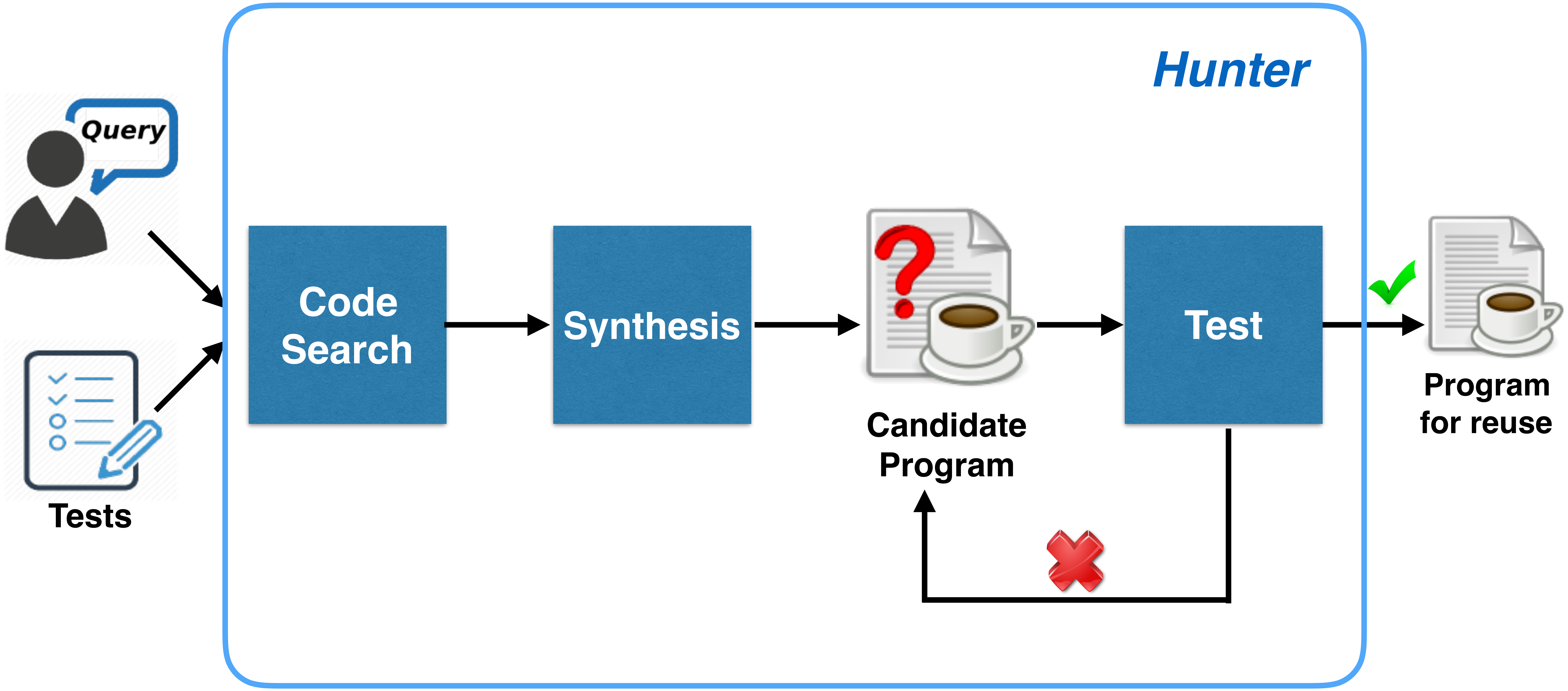
Hunter



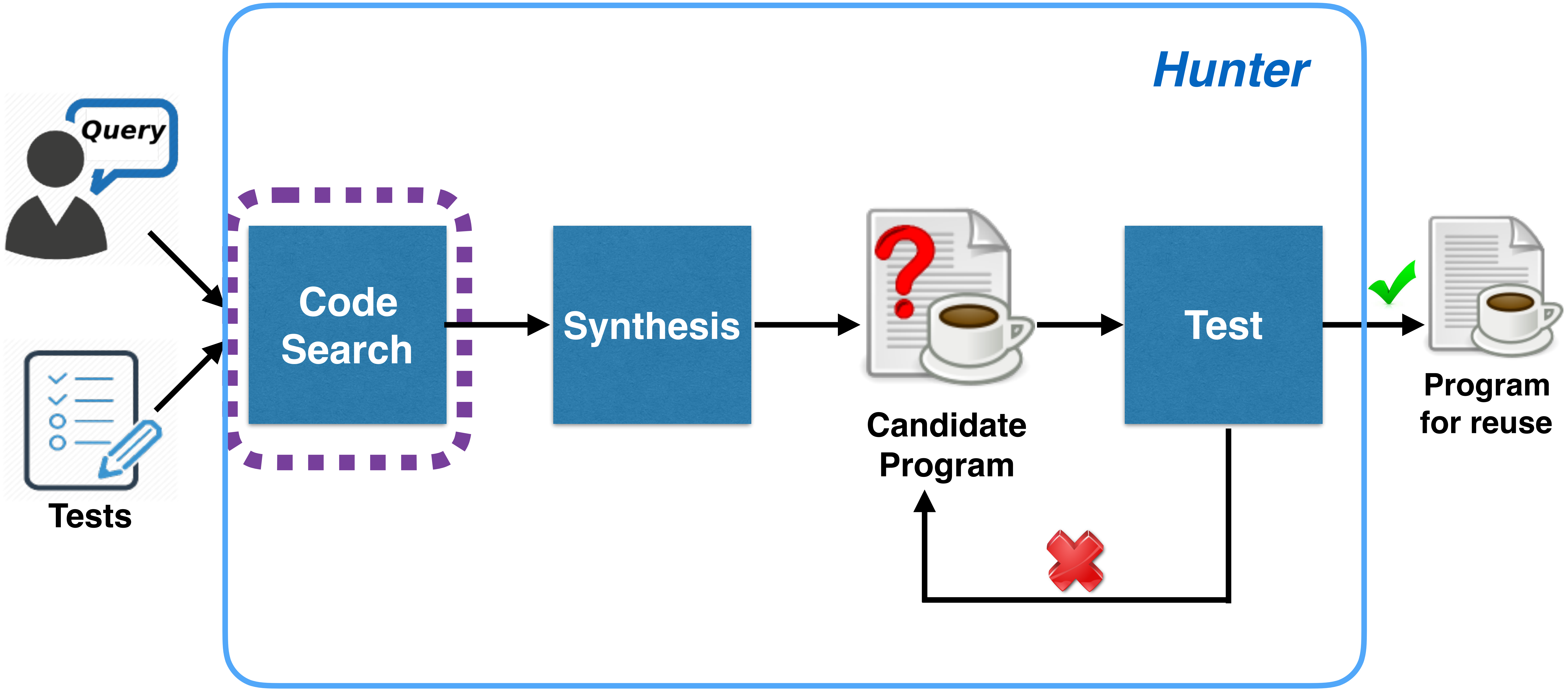
Found code

Wrapper code

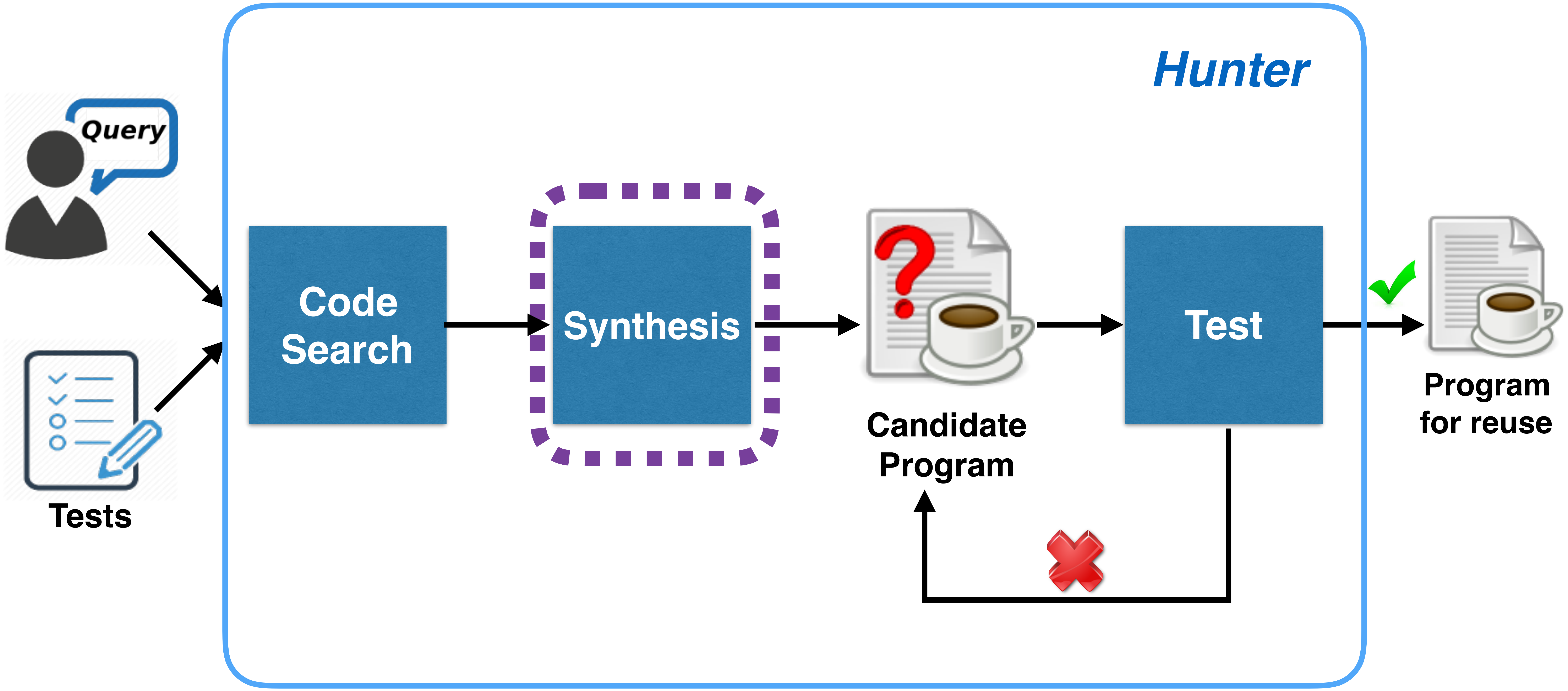
Hunter architecture



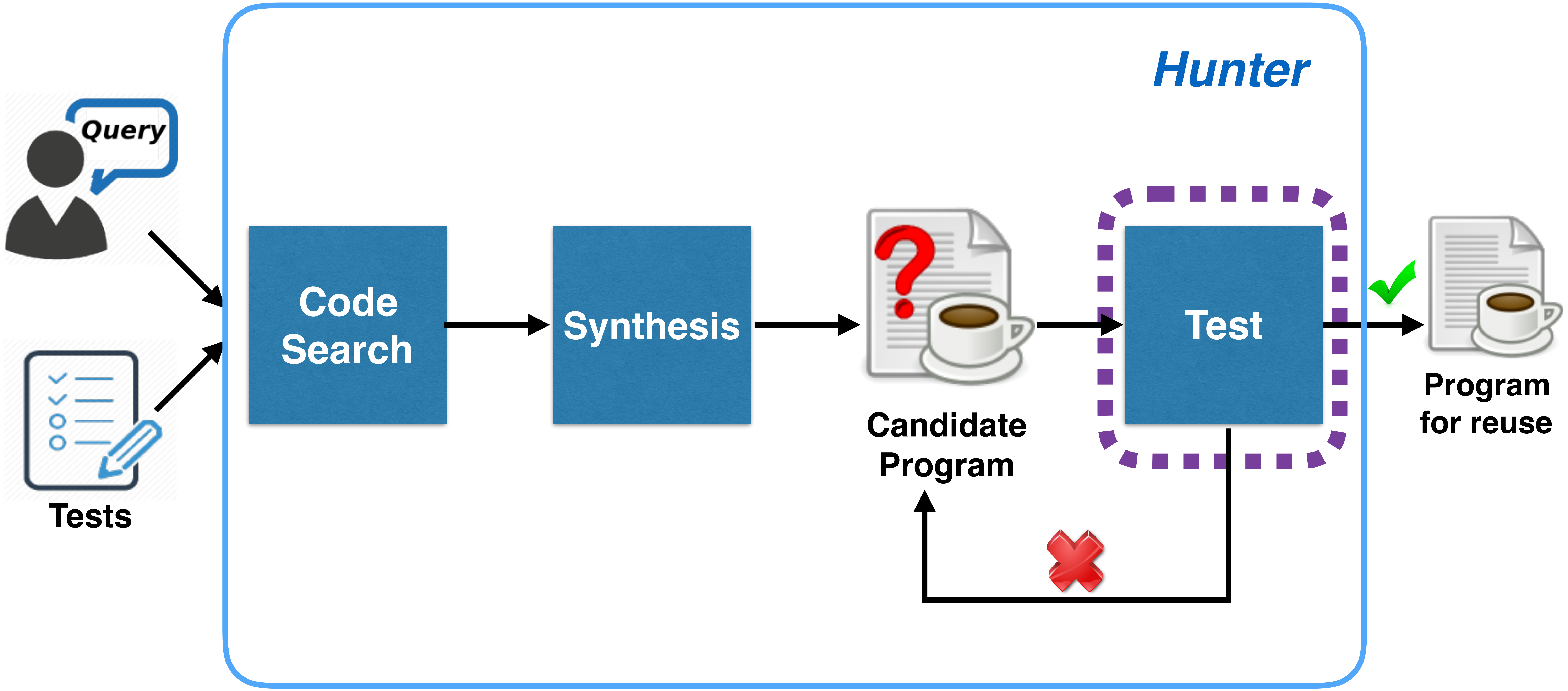
Hunter architecture



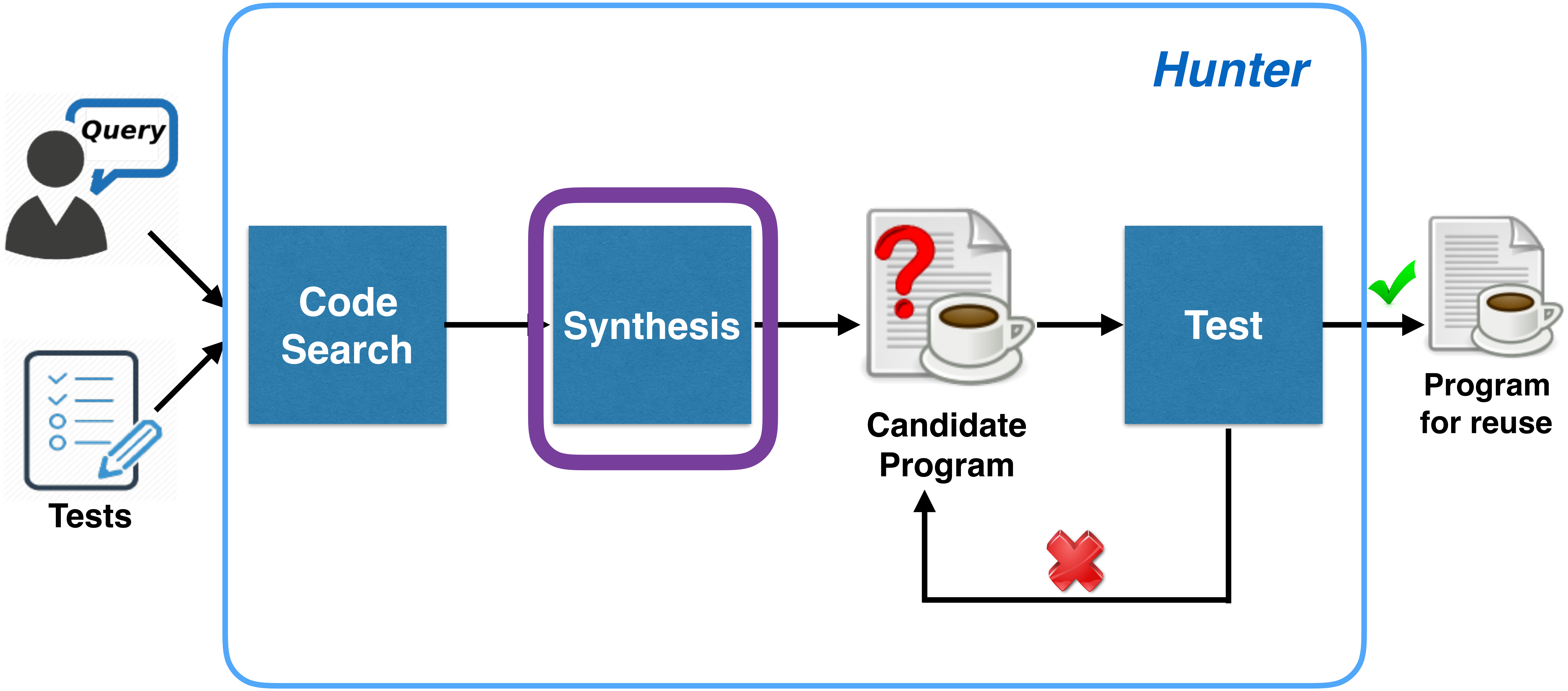
Hunter architecture



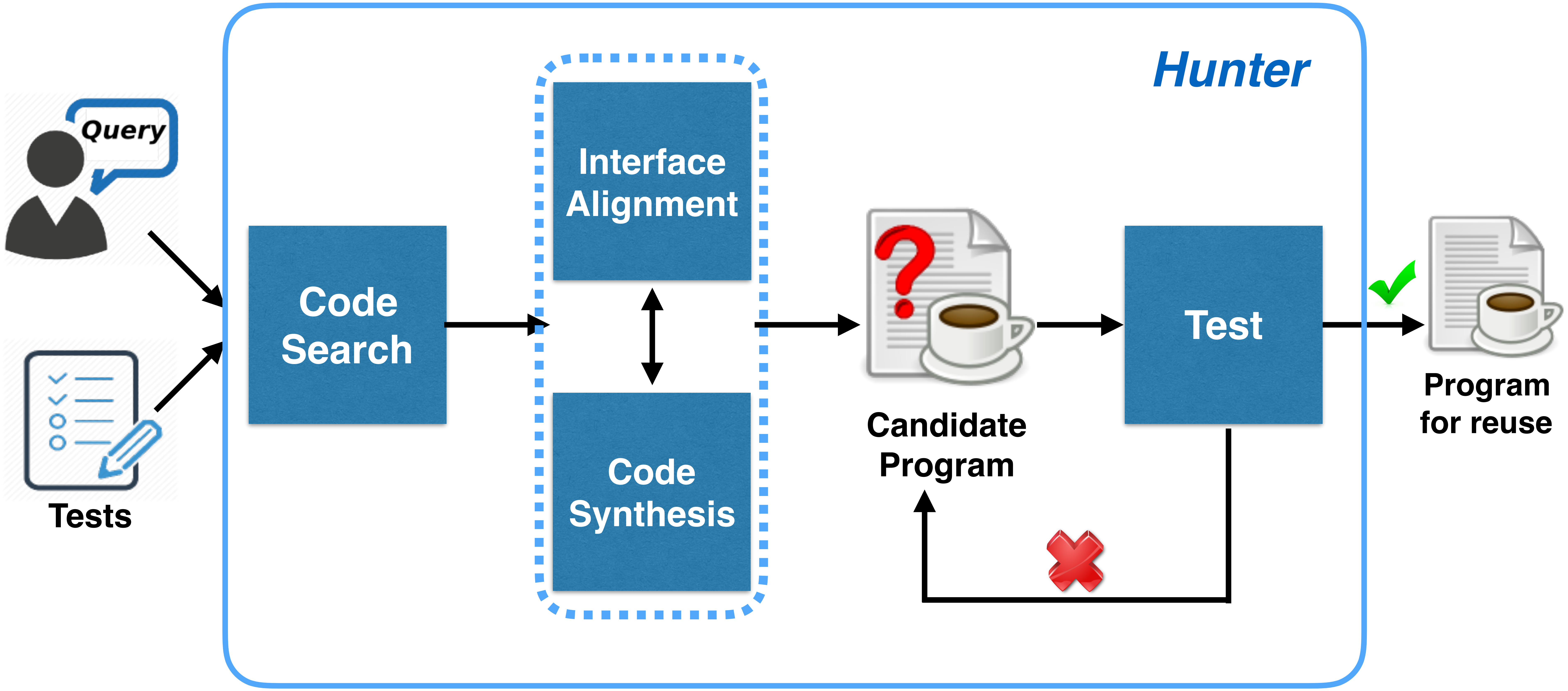
Hunter architecture



Hunter architecture



Hunter architecture



Interface alignment

- Generate reuse scheme by mapping arguments and return value from candidates to the desired signature

desired: **long foo(Point point, int number)**

candidate: **long bar(int num, MyPoint point)**

Code synthesis

- Synthesize **wrapper** to invoke candidates properly
- Handle type conversions by building succinct graph representation and performing reachability analysis

Code synthesis

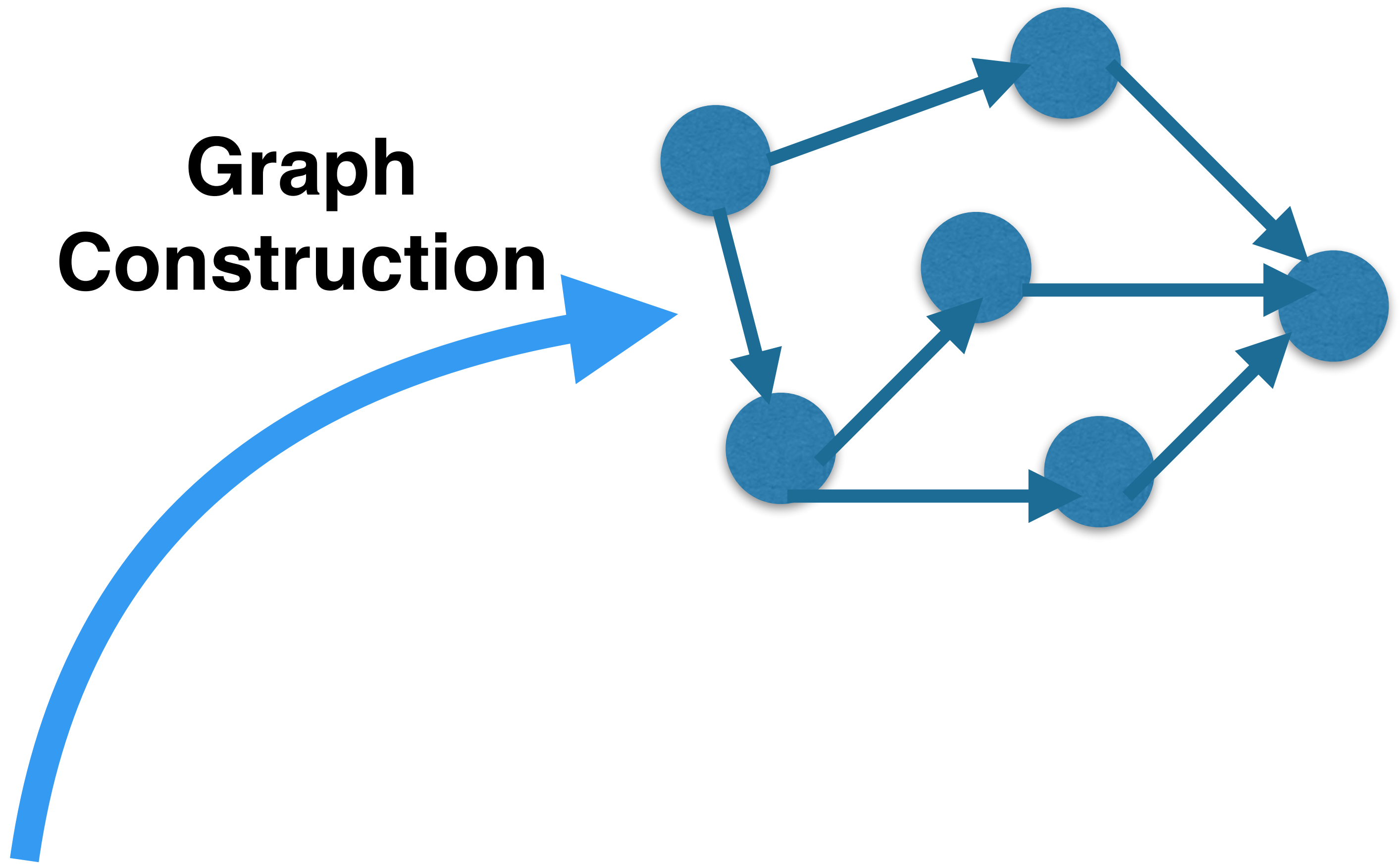
- Synthesize **wrapper** to invoke candidates properly
- Handle type conversions by building succinct graph representation and performing reachability analysis

```
class Point {
   int x; int y;
   Point(int _x, int _y);
}
class MyPoint {
   int x; int y;
   int getX();
   int getY();
}
```

Code synthesis

- Synthesize **wrapper** to invoke candidates properly
- Handle type conversions by building succinct graph representation and performing reachability analysis

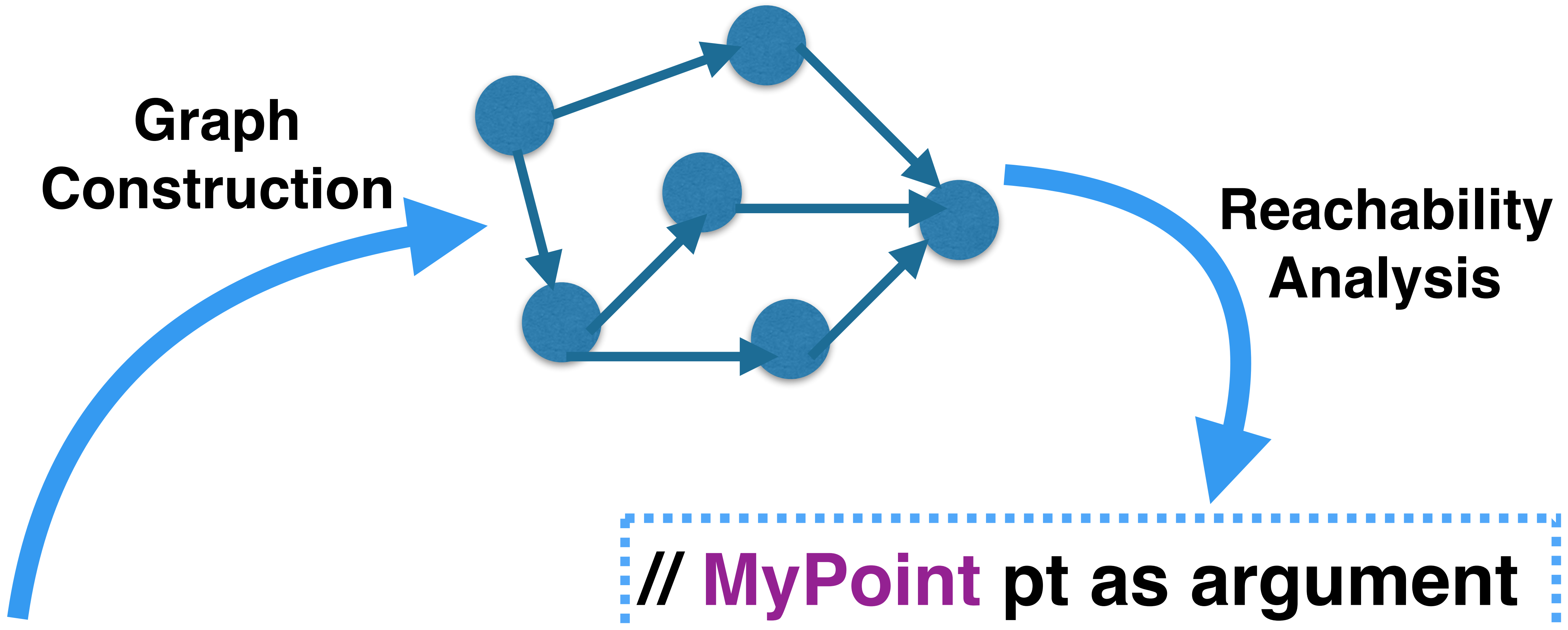
```
class Point {
    int x; int y;
    Point(int _x, int _y);
}
class MyPoint {
    int x; int y;
    int getX();
    int getY();
}
```



Code synthesis

- Synthesize **wrapper** to invoke candidates properly
- Handle type conversions by building succinct graph representation and performing reachability analysis

```
class Point {
   int x; int y;
   Point(int _x, int _y);
}
class MyPoint {
   int x; int y;
   int getX();
   int getY();
}
```




```
// MyPoint pt as argument
int x = pt.getX();
int y = pt.getY();
Point p = new Point(x, y);
```

Who can Program Synthesis help?

Who can Program Synthesis help?

- Are we trying to replace programmers? **No!**
  - We want to make programmers life easier
  - Automating tedious and repetitive tasks

Who can Program Synthesis help?

- Are we trying to replace programmers? **No!**
  - We want to make programmers life easier
  - Automating tedious and repetitive tasks
- 99% of computer users **cannot program**!
  - They struggle with simple repetitive tasks
  - Help non-CS people to automate their daily tasks

Outline

Code Reuse

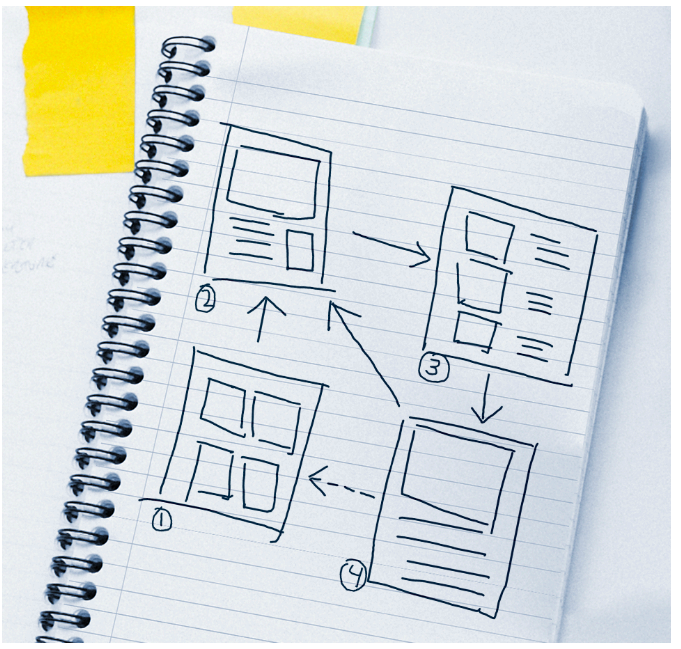
FSE’16

Complex Java APIs

POPL’17

Program Sketching

PLDI’05



CEGIS

PLDI’08

SyGuS

FMCAD’13

SyGuS
Syntax-Guided Synthesis

Outline

Code Reuse

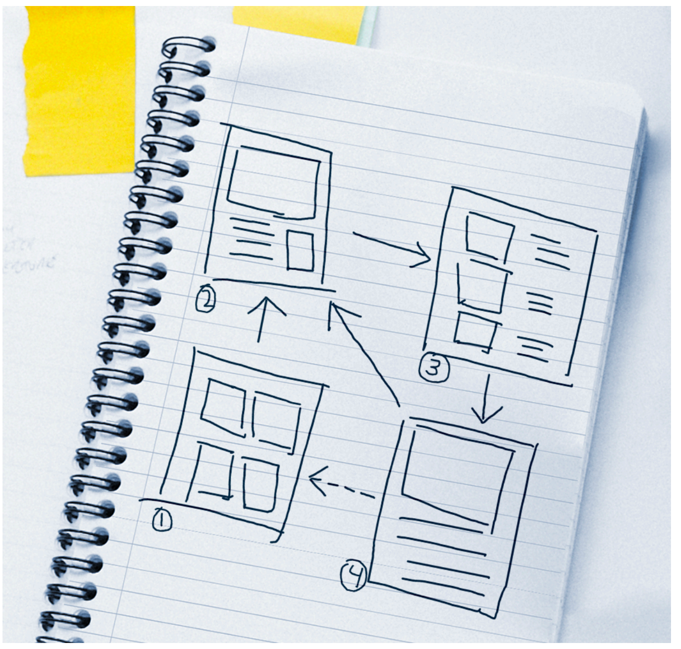
FSE’16

Complex Java APIs

POPL’17

Program Sketching

PLDI’05



CEGIS

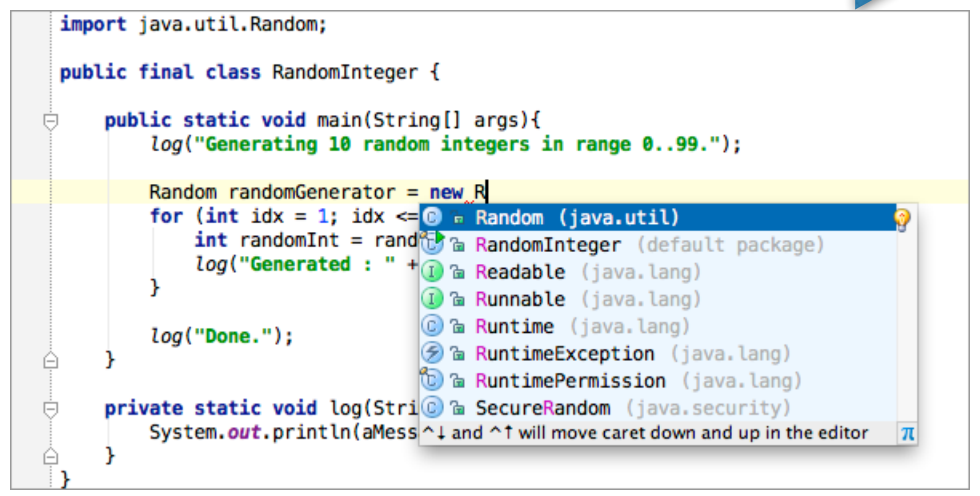
PLDI’08

SyGuS

FMCAD’13

SyGuS
Syntax-Guided Synthesis

API exploration can be tedious!



Send HTTP request
Compute GCD
Rotate an image

Programmers spend a lot of effort
learning APIs!

Synthesizing programs with APIs

- Automatically synthesize an HTTP POST request:

```
import com.mashape.unirest.*;

String sendHttpPost(String url, String body) {




}
```

Click!

Synthesizing programs with APIs

- Automatically synthesize an HTTP POST request:

```
import com.mashape.unirest.*;

String sendHttpPost(String url, String body) {

  HttpRequestWithBody req = post(url);
  RequestBodyEntity ent = req.body(body);
  BaseRequest breq = ent;
  HttpResponse resp = breq.asString();
  String result = resp.getStringBody();
  return result;

}
```

How to find the correct program?

How to find the correct program?

Use **Petri net reachability** analysis to look for well-typed programs of the desired type

How to find the correct program?

Use **Petri net reachability** analysis to look for well-typed programs of the desired type

- Model relationships between components using Petri nets

How to find the correct program?

Use **Petri net reachability** analysis to look for well-typed programs of the desired type

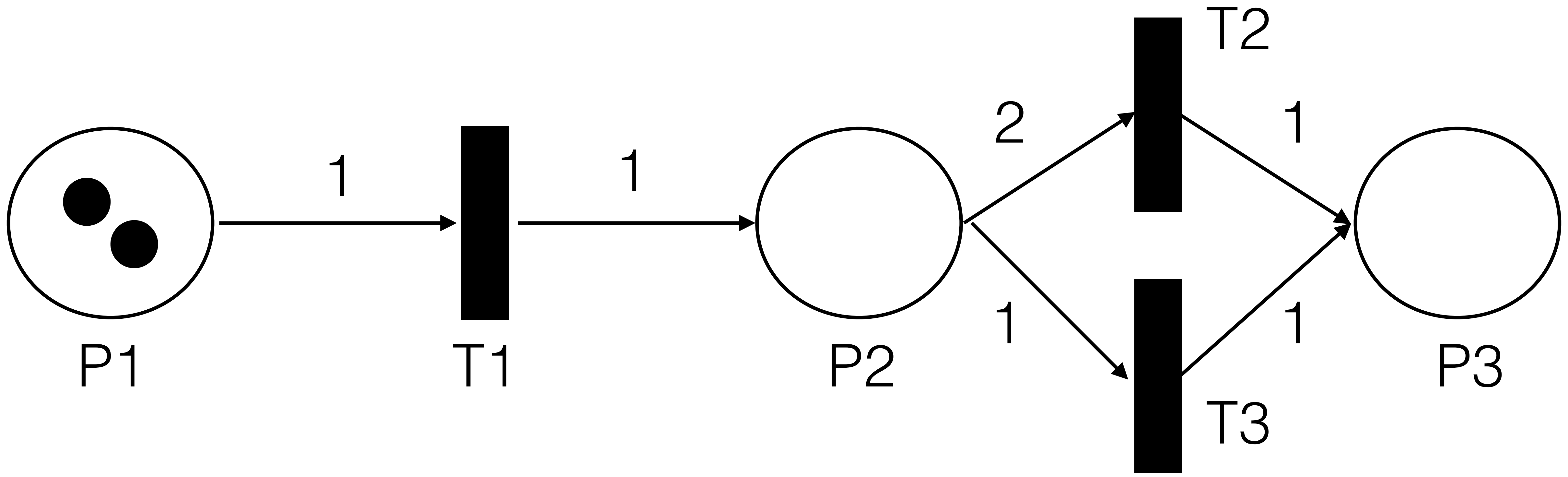
- Model relationships between components using Petri nets
- Use type signature of desired method to mark initial and target configurations

How to find the correct program?

Use **Petri net reachability** analysis to look for well-typed programs of the desired type

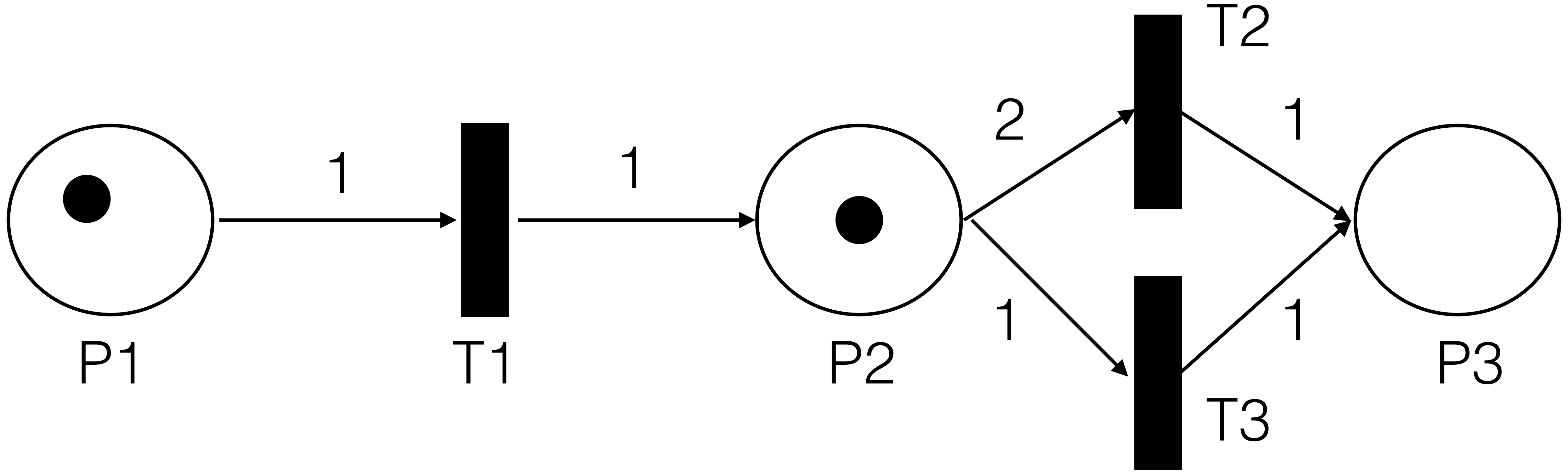
- Model relationships between components using Petri nets
- Use type signature of desired method to mark initial and target configurations
- Perform reachability analysis to find valid sequences of method calls

Petri nets in a nutshell



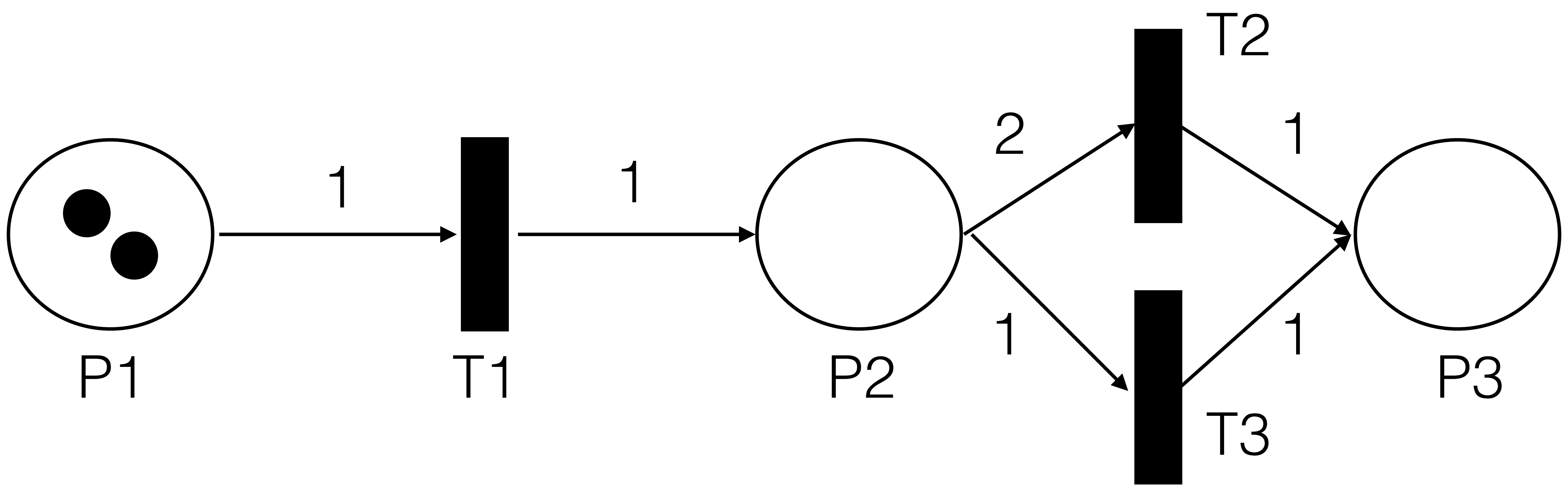
- Petri net is a generalized graph with two kinds of nodes: **places** and **transitions**
- Each place contains zero or more tokens; edges are labeled with a number (of tokens)
- A transition T can fire if, for each edge (p,T) with label n, place p contains at least n tokens
- Firing a transition T consumes (resp. produces) the indicated number of tokens at the source (resp. target) nodes

Petri nets in a nutshell



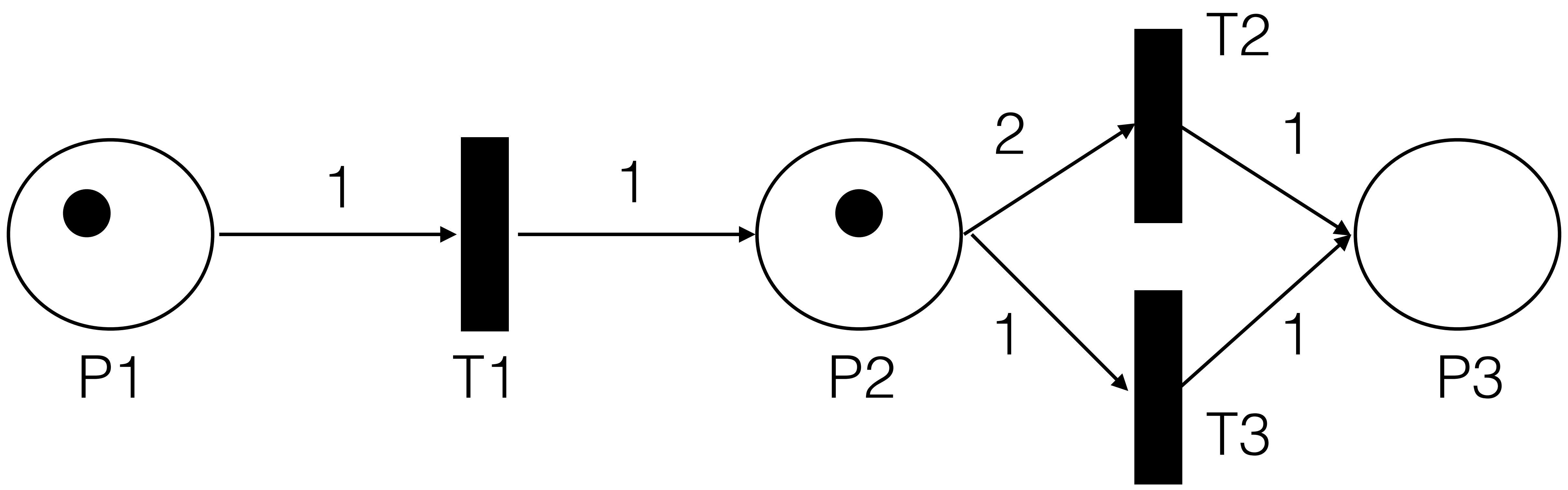
- Petri net is a generalized graph with two kinds of nodes: **places** and **transitions**
- Each place contains zero or more tokens; edges are labeled with a number (of tokens)
- A transition T can fire if, for each edge (p,T) with label n, place p contains at least n tokens
- Firing a transition T consumes (resp. produces) the indicated number of tokens at the source (resp. target) nodes

Reachability problem in Petri nets



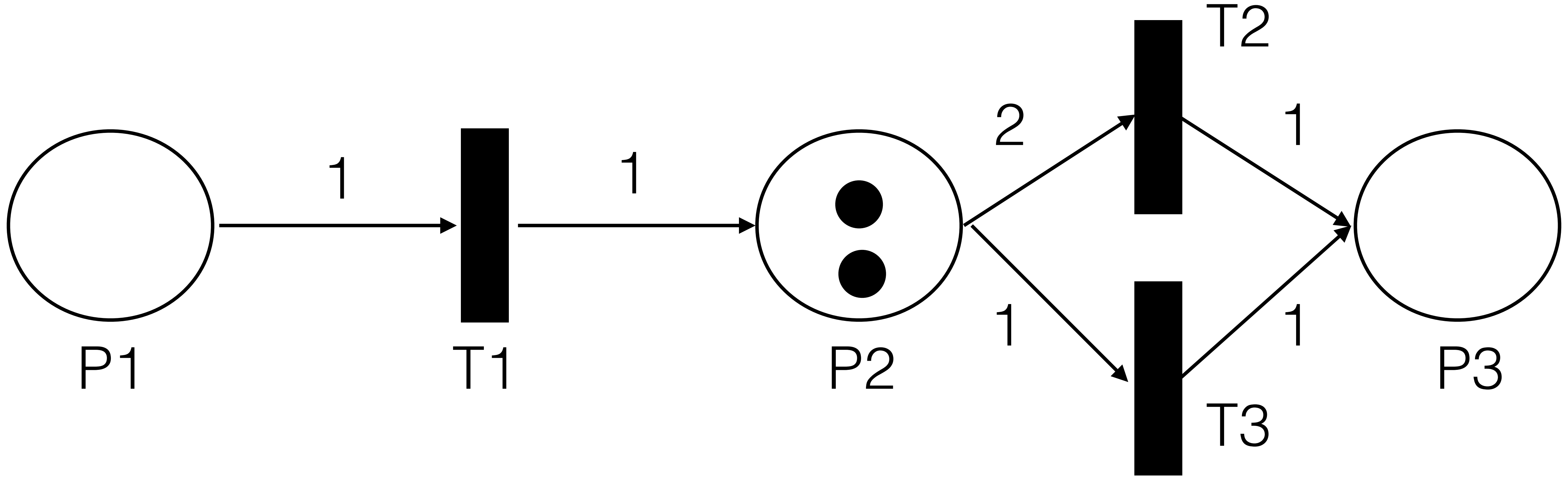
- Reachability problem: Given a Petri net with initial marking M and a target marking M′, is it possible to obtain M’ by firing a sequence of transitions?
- Example: Consider marking M′ : [P1→0,P2→0,P3→1].
- This marking is reachable, and accepting run is T1,T1,T2.

Reachability problem in Petri nets



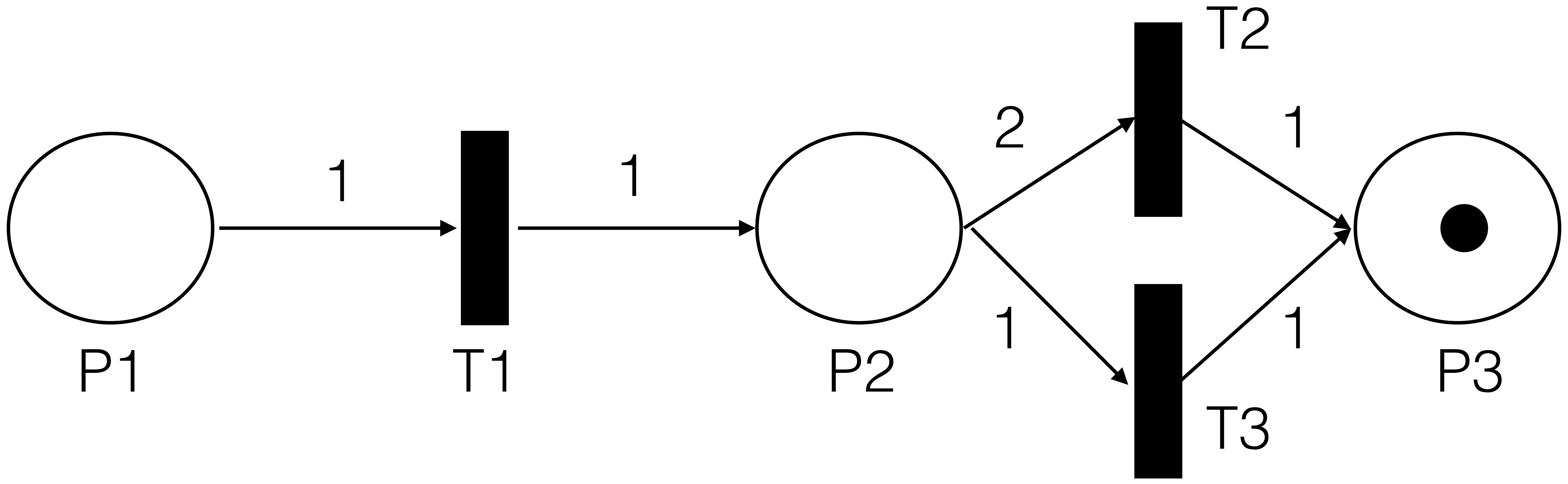
- Reachability problem: Given a Petri net with initial marking M and a target marking M′, is it possible to obtain M’ by firing a sequence of transitions?
- Example: Consider marking M′ : [P1→0,P2→0,P3→1].
- This marking is reachable, and accepting run is T1,T1,T2.

Reachability problem in Petri nets



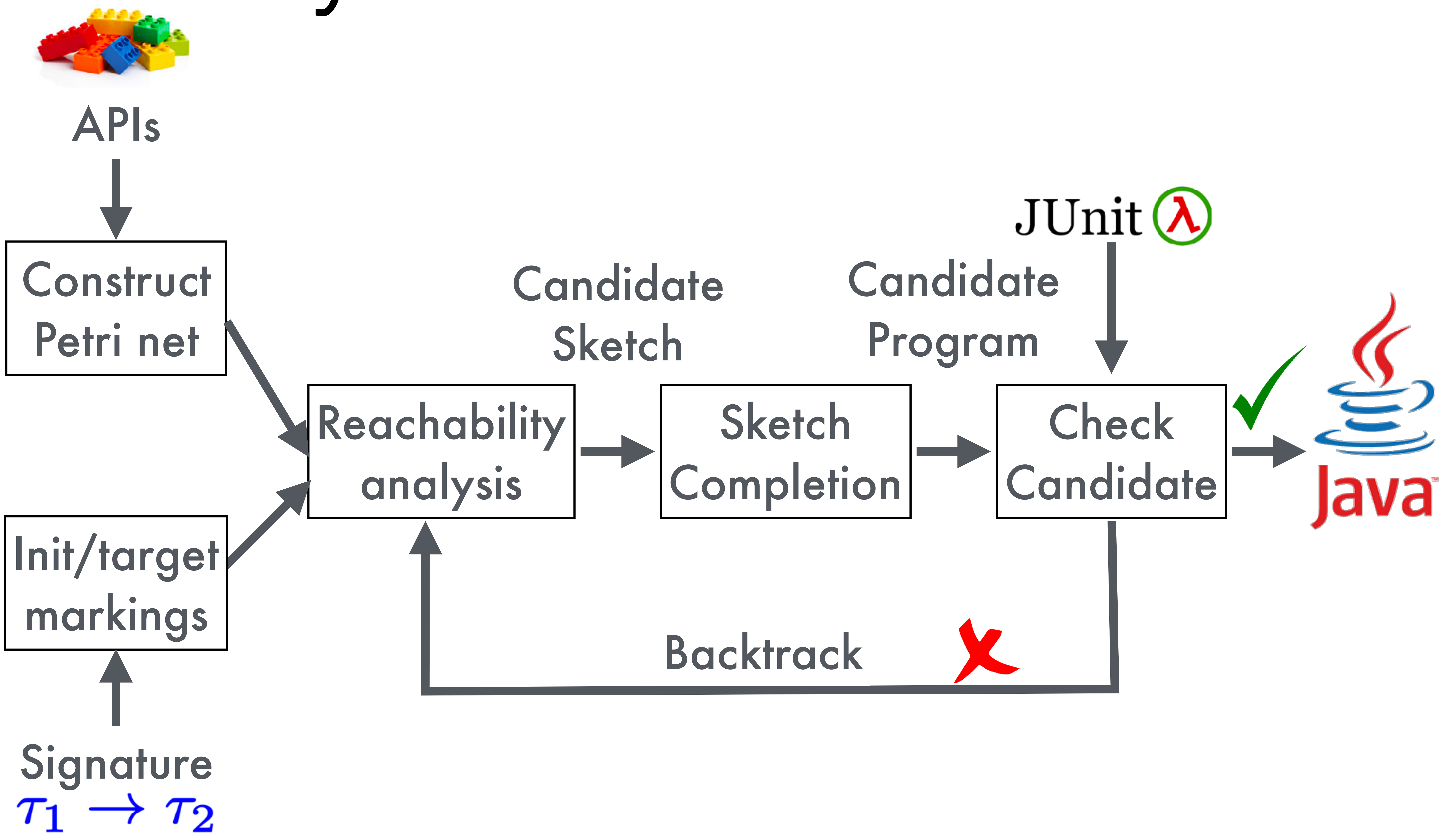
- Reachability problem: Given a Petri net with initial marking M and a target marking M′, is it possible to obtain M’ by firing a sequence of transitions?
- Example: Consider marking M′ : [P1→0,P2→0,P3→1].
- This marking is reachable, and accepting run is T1,T1,T2.

Reachability problem in Petri nets

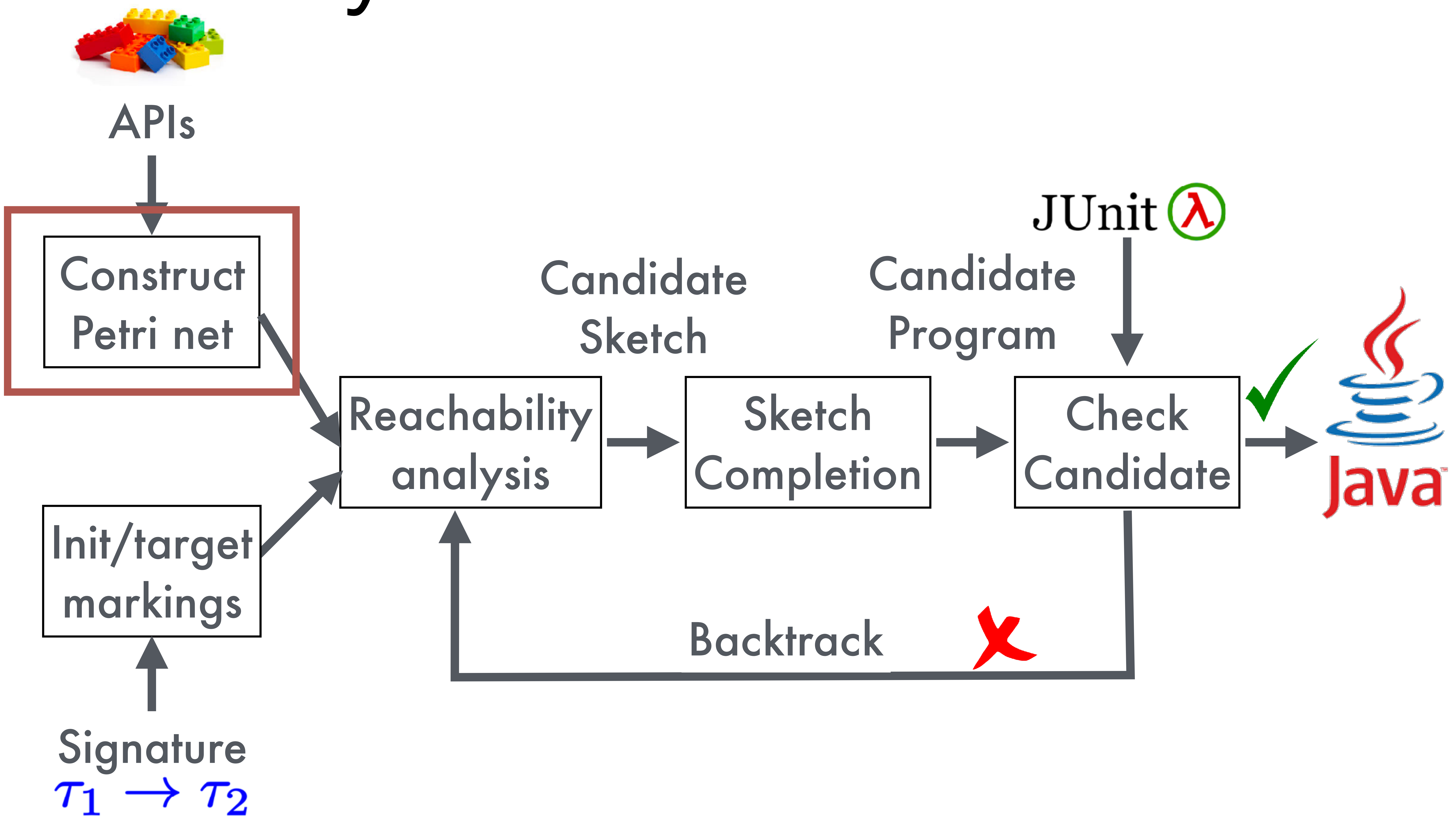


- Reachability problem: Given a Petri net with initial marking M and a target marking M′, is it possible to obtain M’ by firing a sequence of transitions?
- Example: Consider marking M′ : [P1→0,P2→0,P3→1].
- This marking is reachable, and accepting run is T1,T1,T2.

SyPet architecture

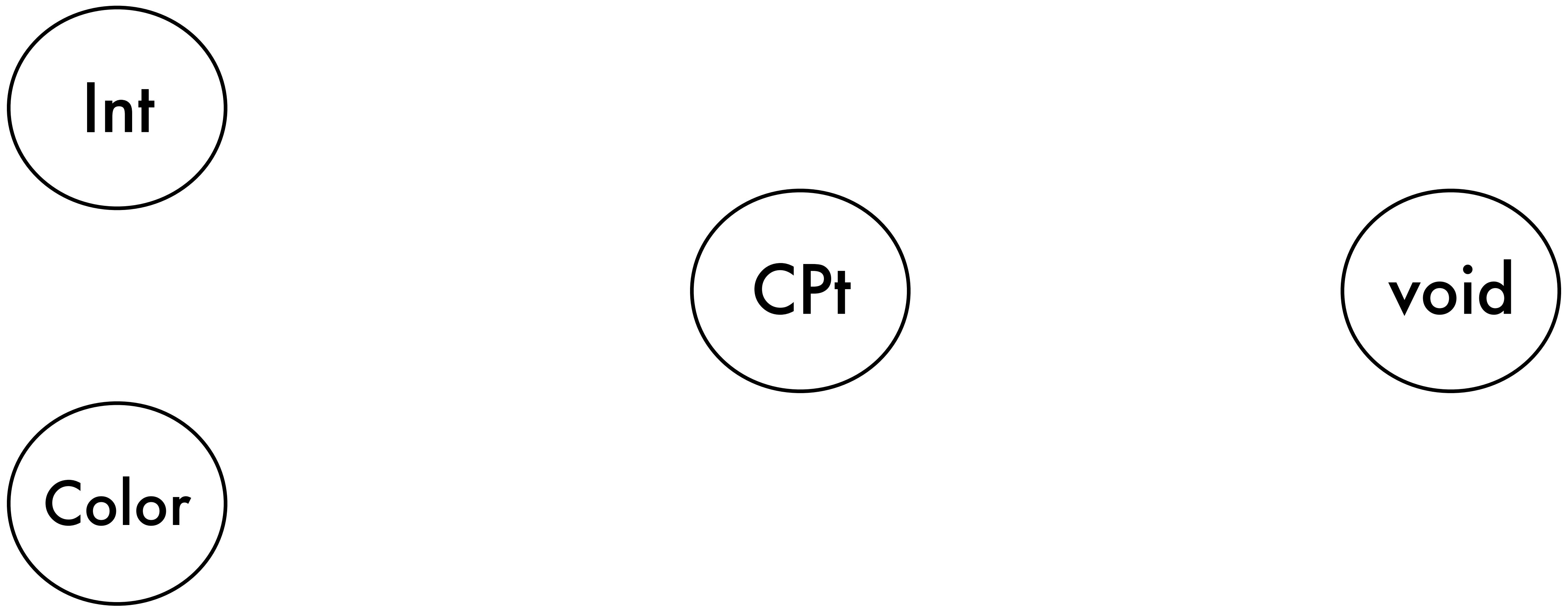


SyPet architecture



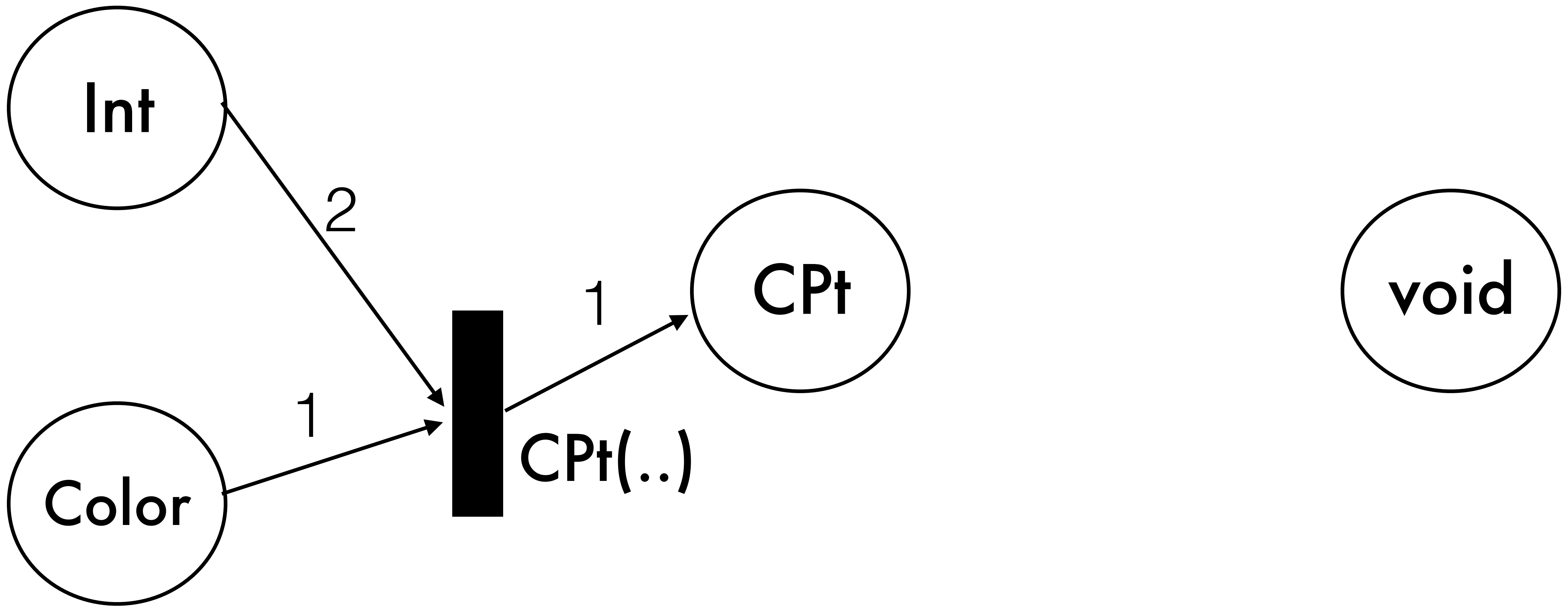
Petri net construction

```
class CPt {
  CPt(Int x, Int y, Color c);
  Int getX();
  void setColor(Color c);
  ...
}
```



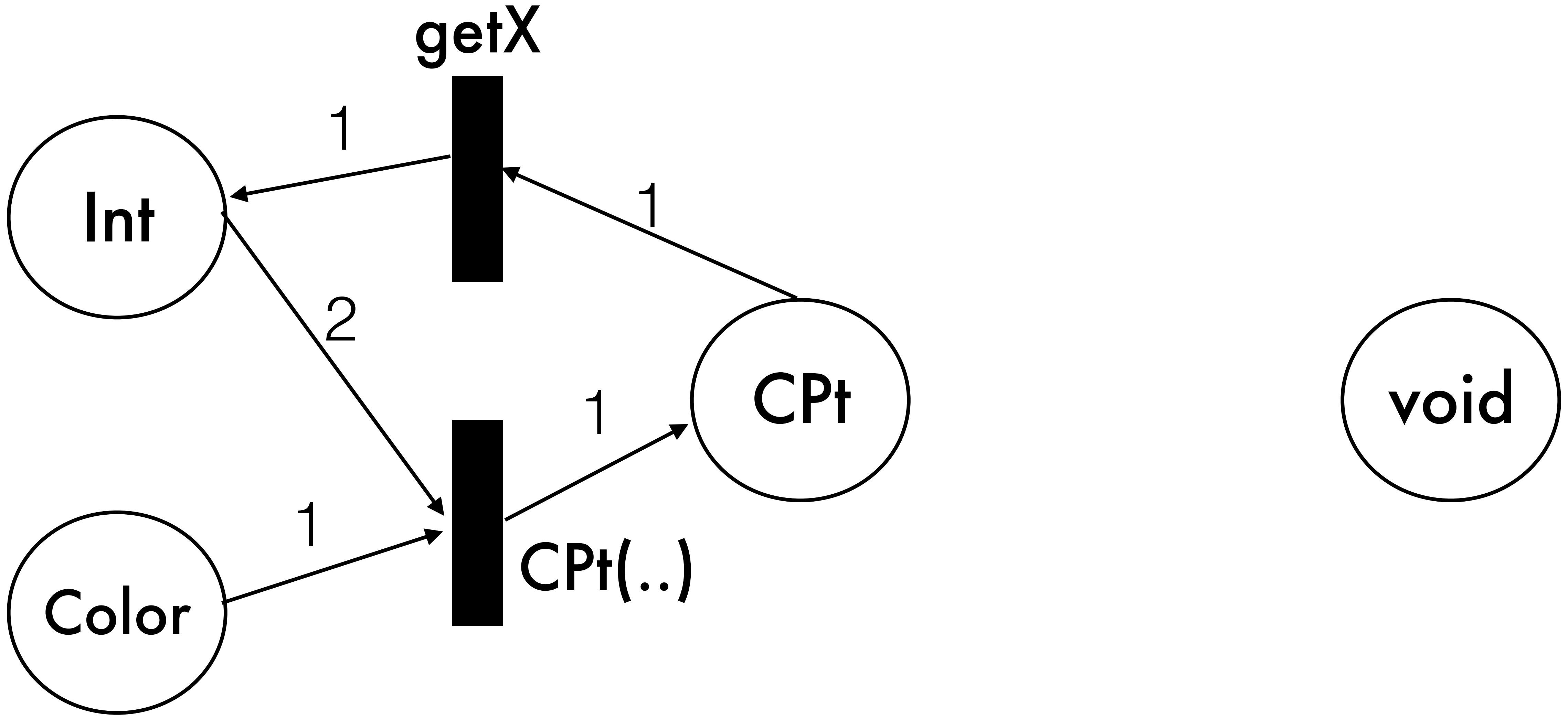
Petri net construction

```
class CPt {
  CPt(Int x, Int y, Color c);
  Int getX();
  void setColor(Color c);
  ...
}
```



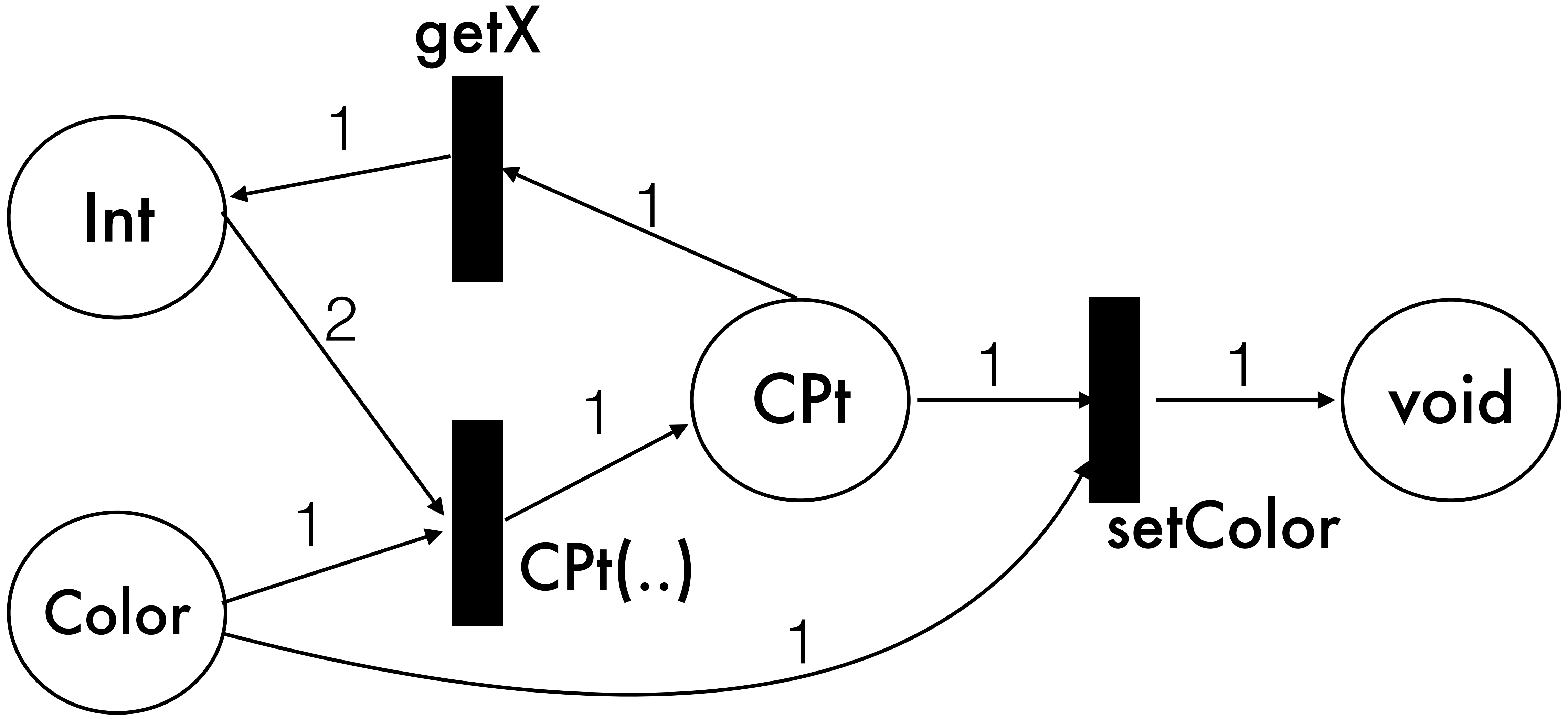
Petri net construction

```
class CPt {
  CPt(Int x, Int y, Color c);
  Int getX();
  void setColor(Color c);
  ...
}
```

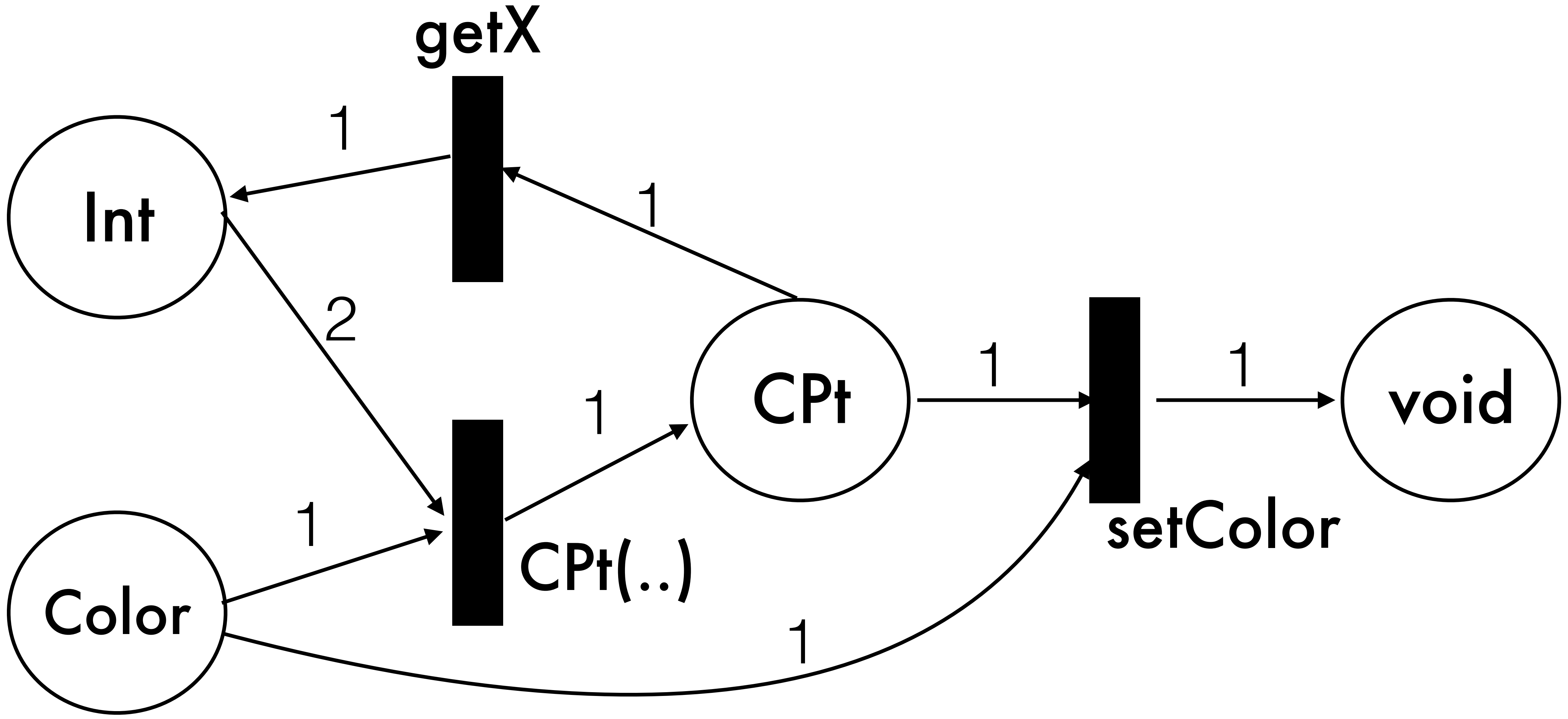


Petri net construction

```
class CPt {
  CPt(Int x, Int y, Color c);
  Int getX();
  void setColor(Color c);
  ...
}
```

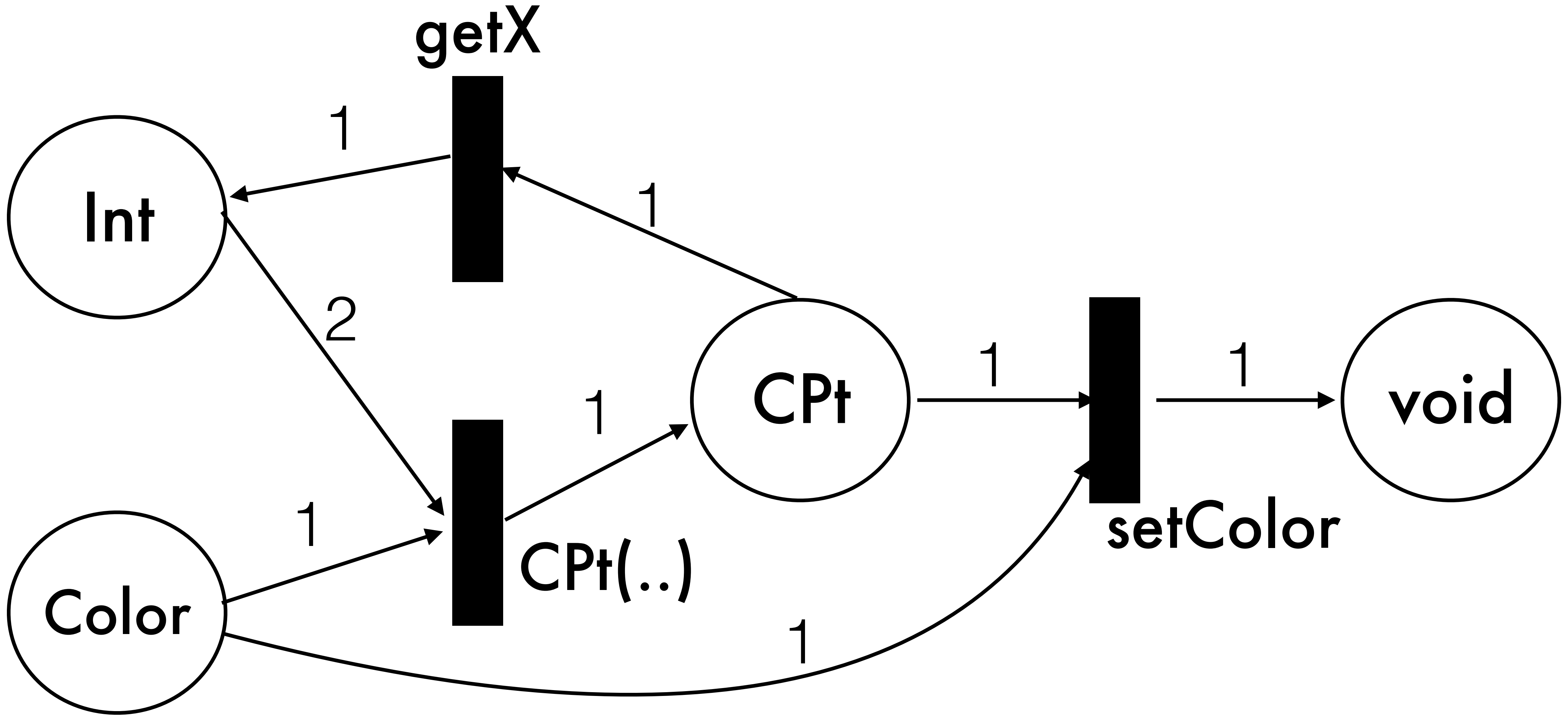


Clone transitions



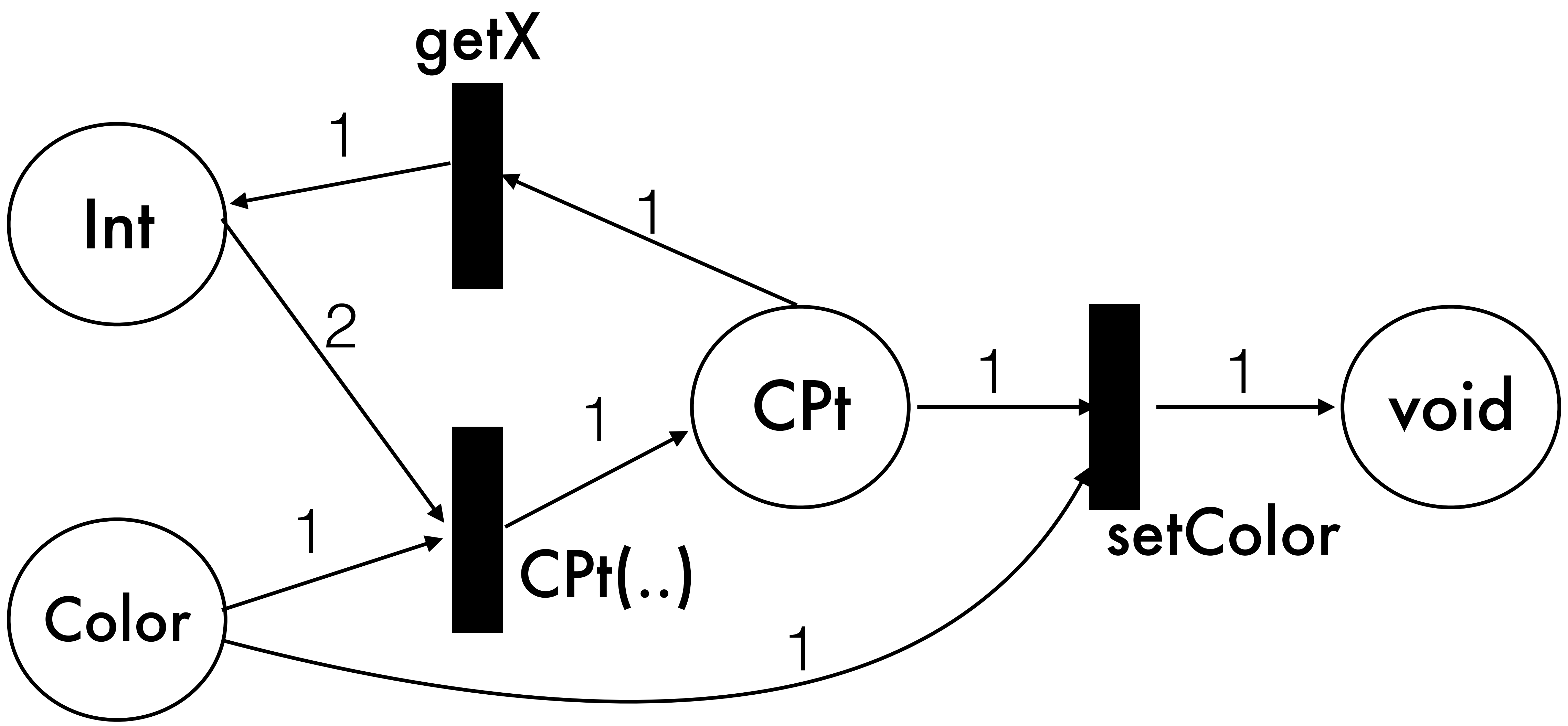
Clone transitions

- Our construction so far views objects as“resources”– every method “consumes” and “produces” objects



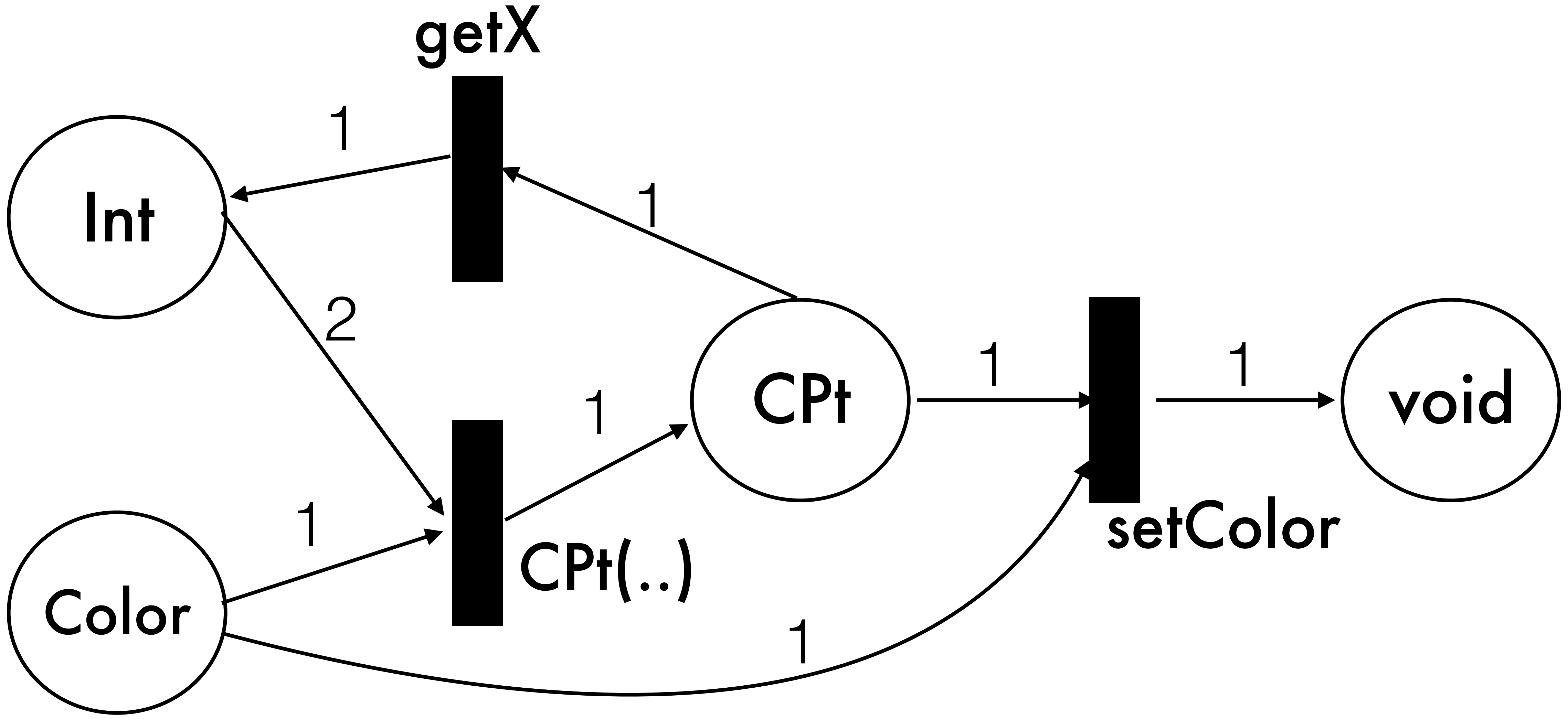
Clone transitions

- Our construction so far views objects as“resources”– every method “consumes” and “produces” objects
- But in conventional languages, we can reuse objects!



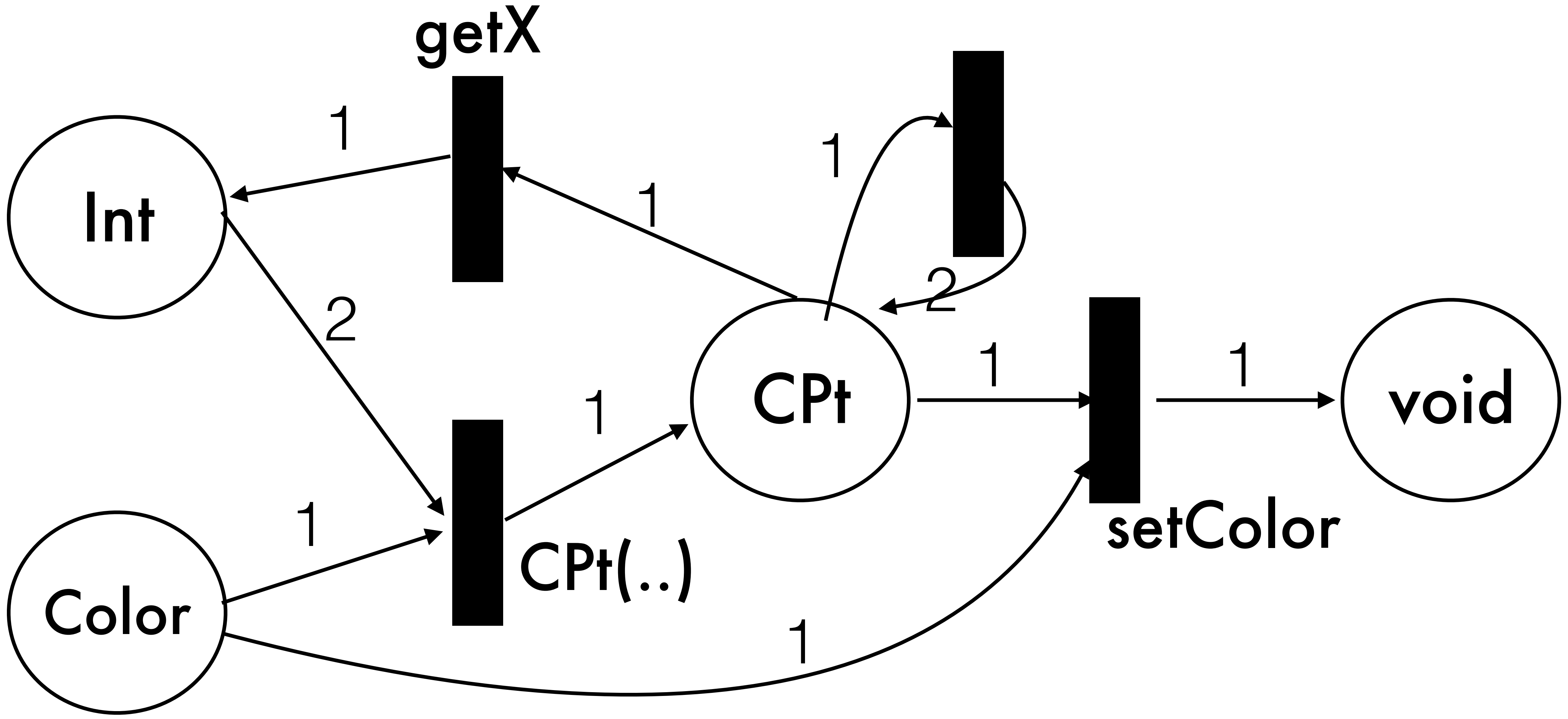
Clone transitions

- Our construction so far views objects as“resources”– every method “consumes” and “produces” objects
- But in conventional languages, we can reuse objects!
- Therefore, augment Petri net model with **clone transitions**



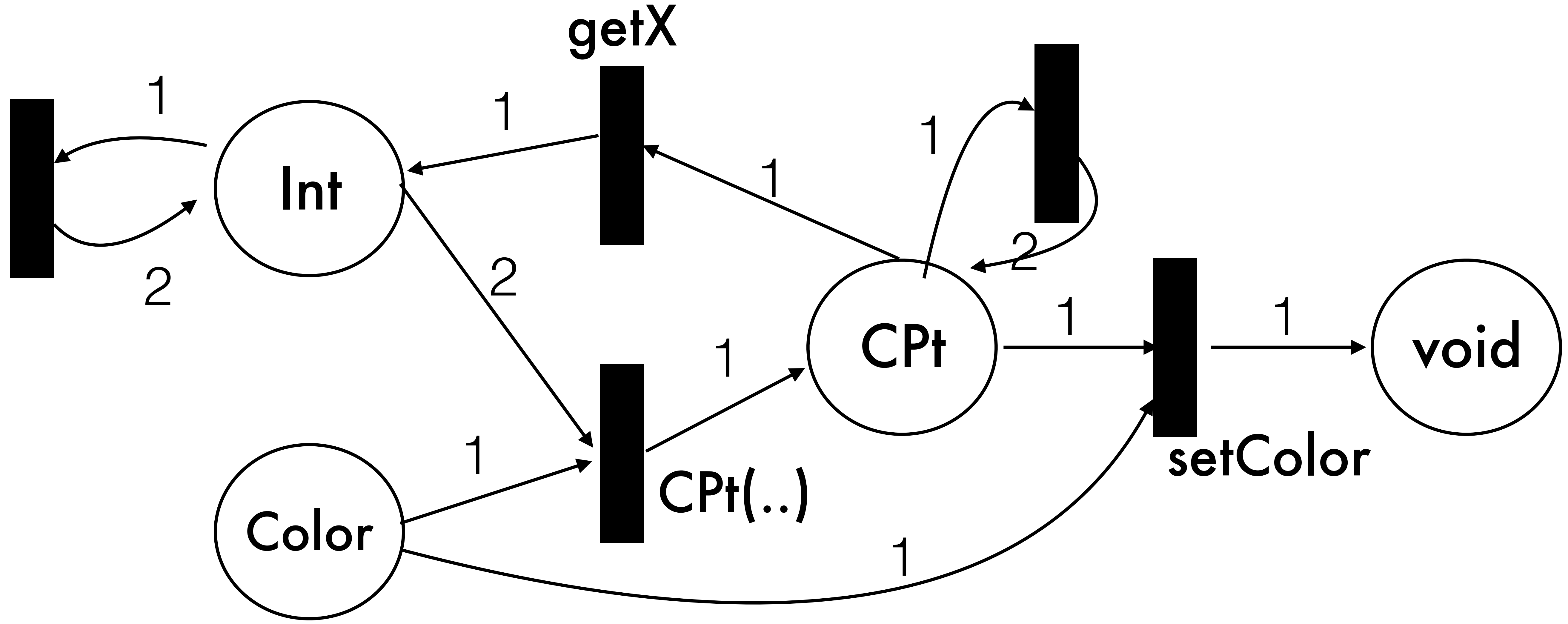
Clone transitions

- Our construction so far views objects as“resources”– every method “consumes” and “produces” objects
- But in conventional languages, we can reuse objects!
- Therefore, augment Petri net model with **clone transitions**



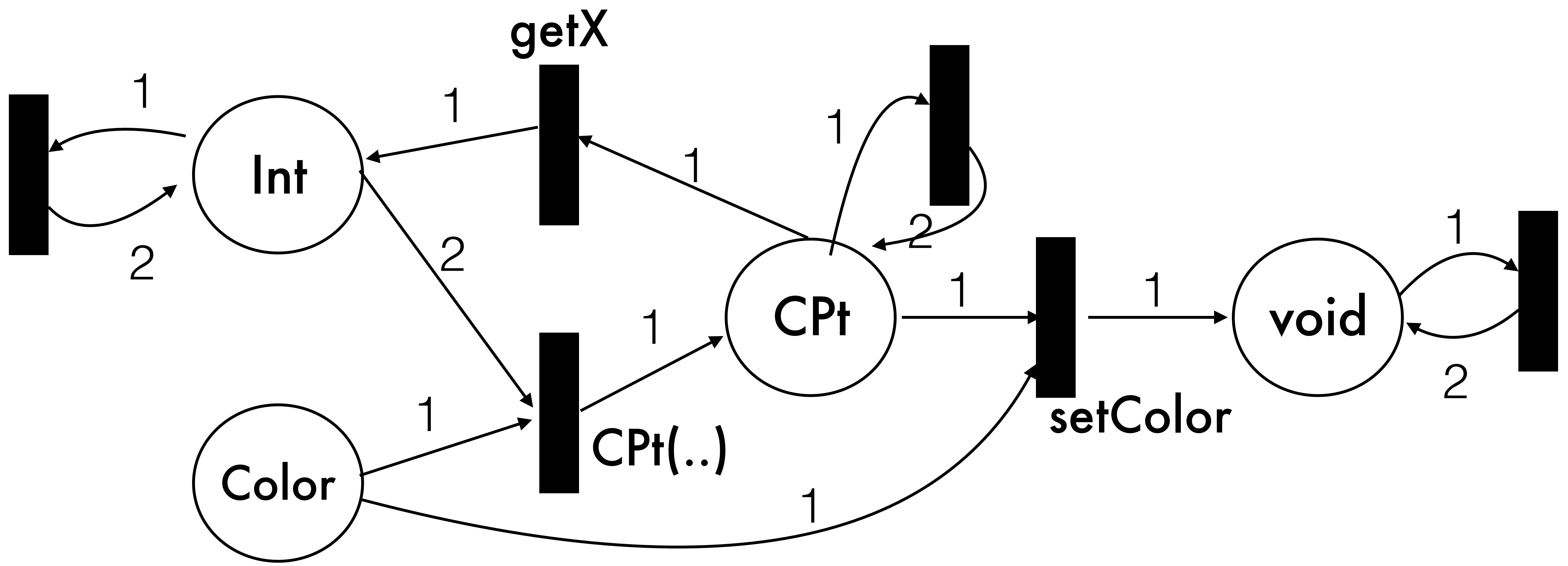
Clone transitions

- Our construction so far views objects as“resources”– every method “consumes” and “produces” objects
- But in conventional languages, we can reuse objects!
- Therefore, augment Petri net model with **clone transitions**



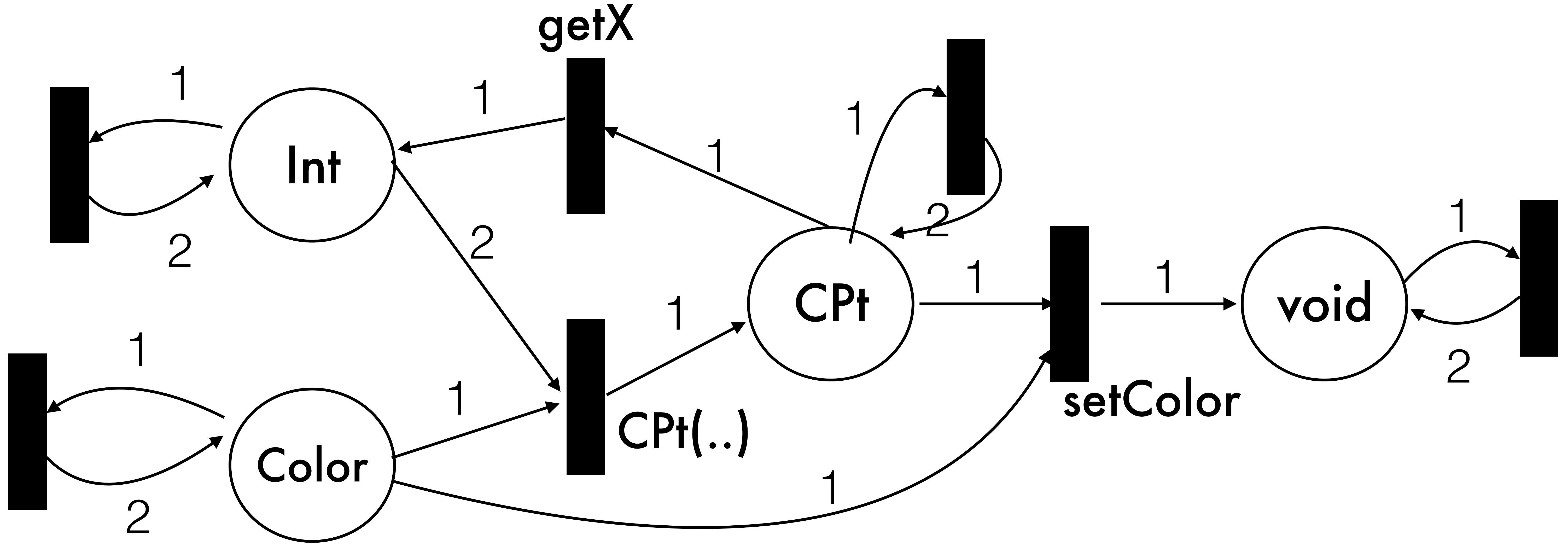
Clone transitions

- Our construction so far views objects as“resources”– every method “consumes” and “produces” objects
- But in conventional languages, we can reuse objects!
- Therefore, augment Petri net model with **clone transitions**

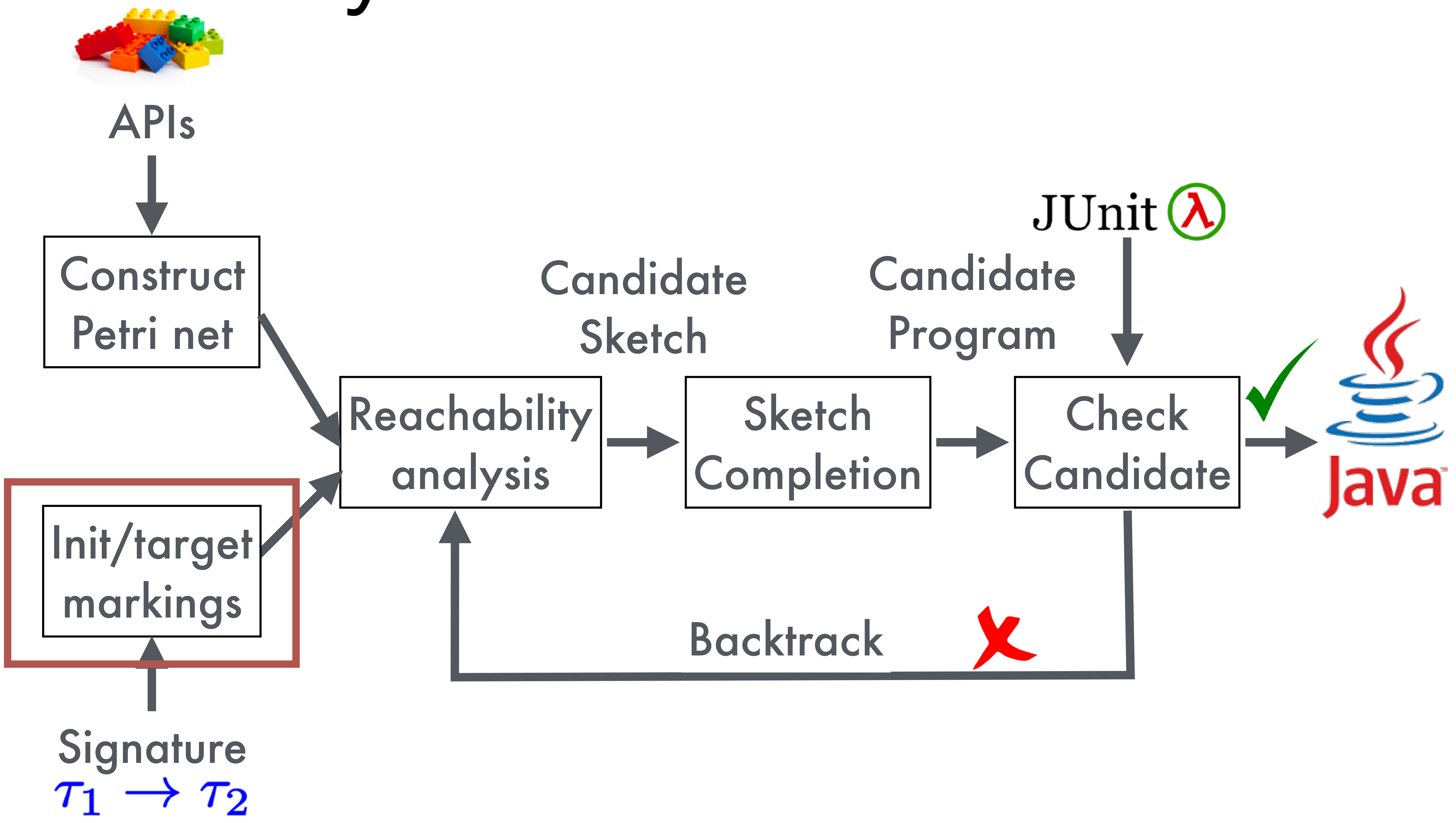


Clone transitions

- Our construction so far views objects as“resources”– every method “consumes” and “produces” objects
- But in conventional languages, we can reuse objects!
- Therefore, augment Petri net model with **clone transitions**

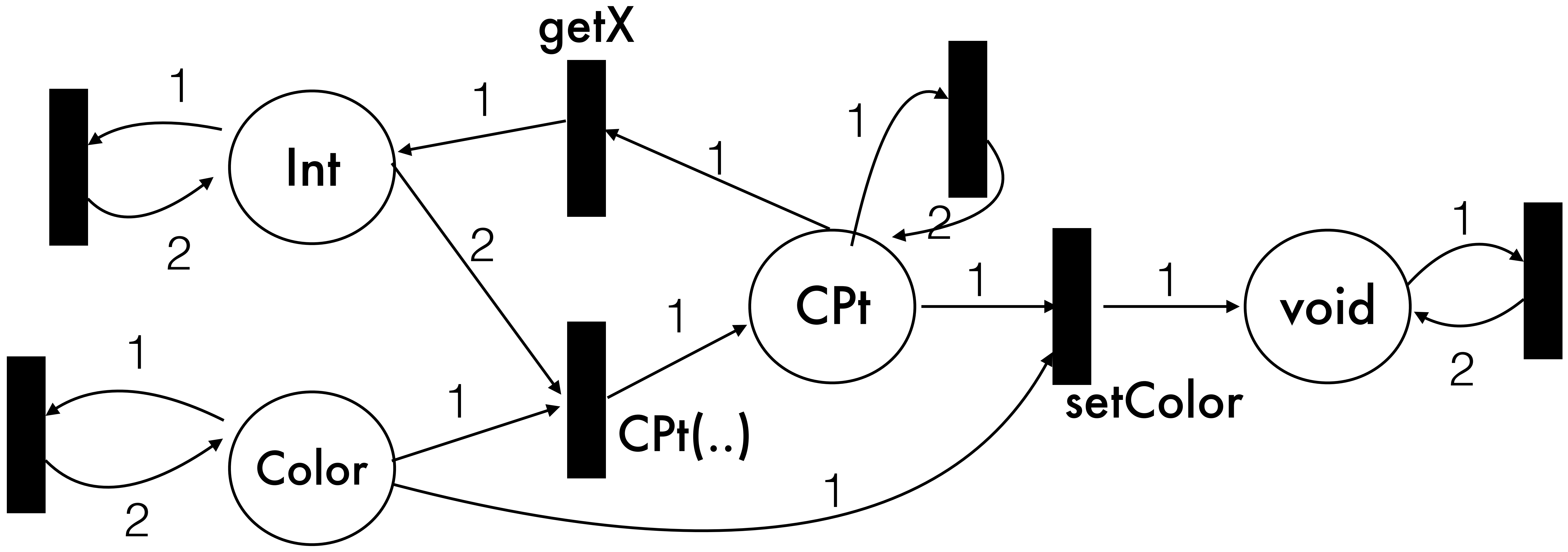


SyPet architecture



Initial and target markings

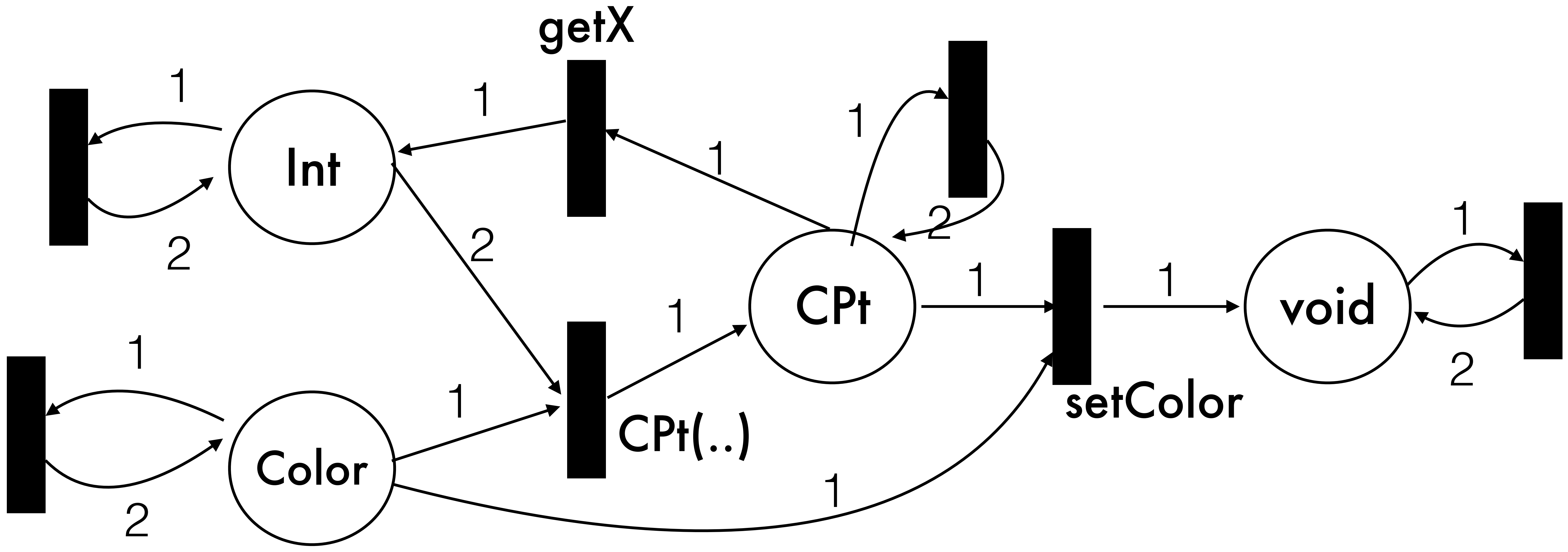
Use signature to determine initial and target markings of Petri net



Initial and target markings

Use signature to determine initial and target markings of Petri net

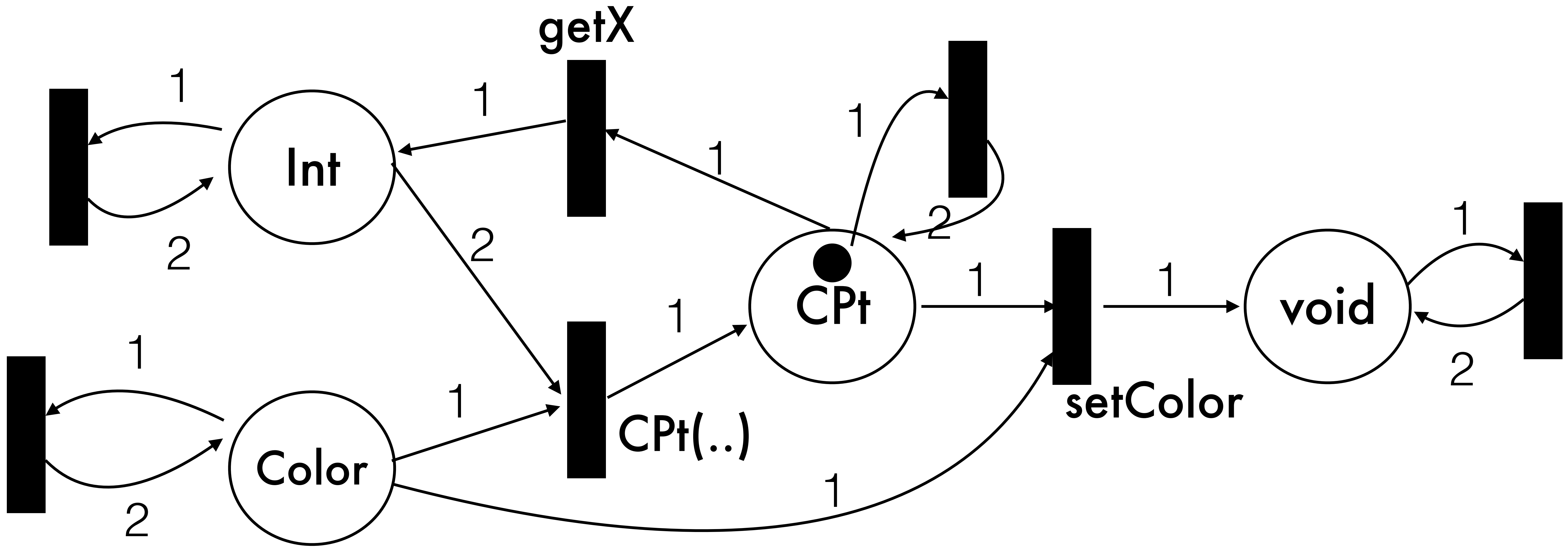
```
CPt shift (CPt p, Int shiftX, Int shiftY)
```



Initial and target markings

Use signature to determine initial and target markings of Petri net

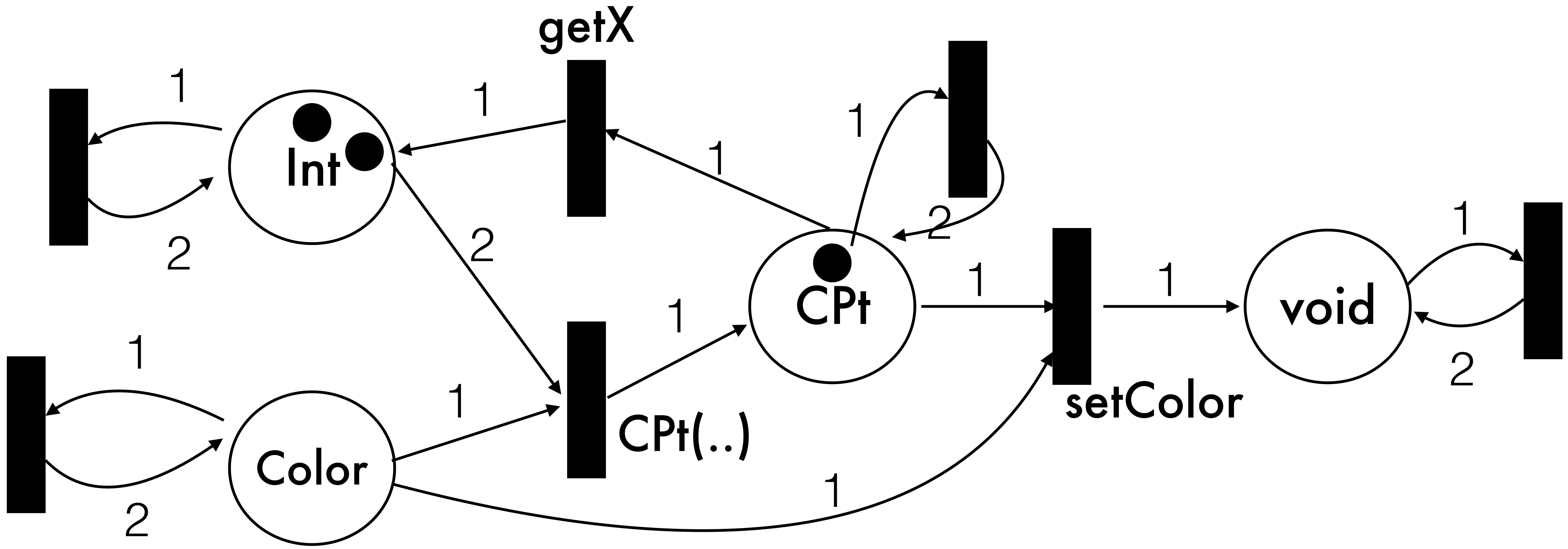
```
CPt shift (CPt p, Int shiftX, Int shiftY)
```



Initial and target markings

Use signature to determine initial and target markings of Petri net

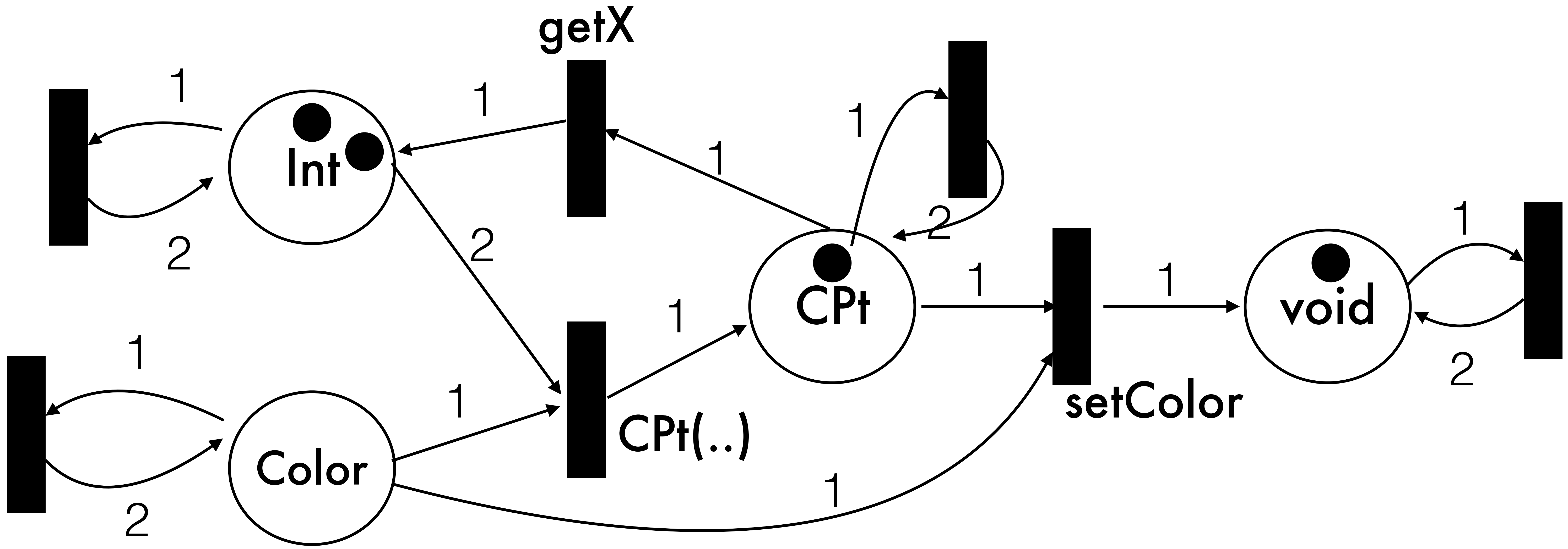
```
CPt shift (CPt p, Int shiftX, Int shiftY)
```



Initial and target markings

Use signature to determine initial and target markings of Petri net

```
CPt shift (CPt p, Int shiftX, Int shiftY)
```

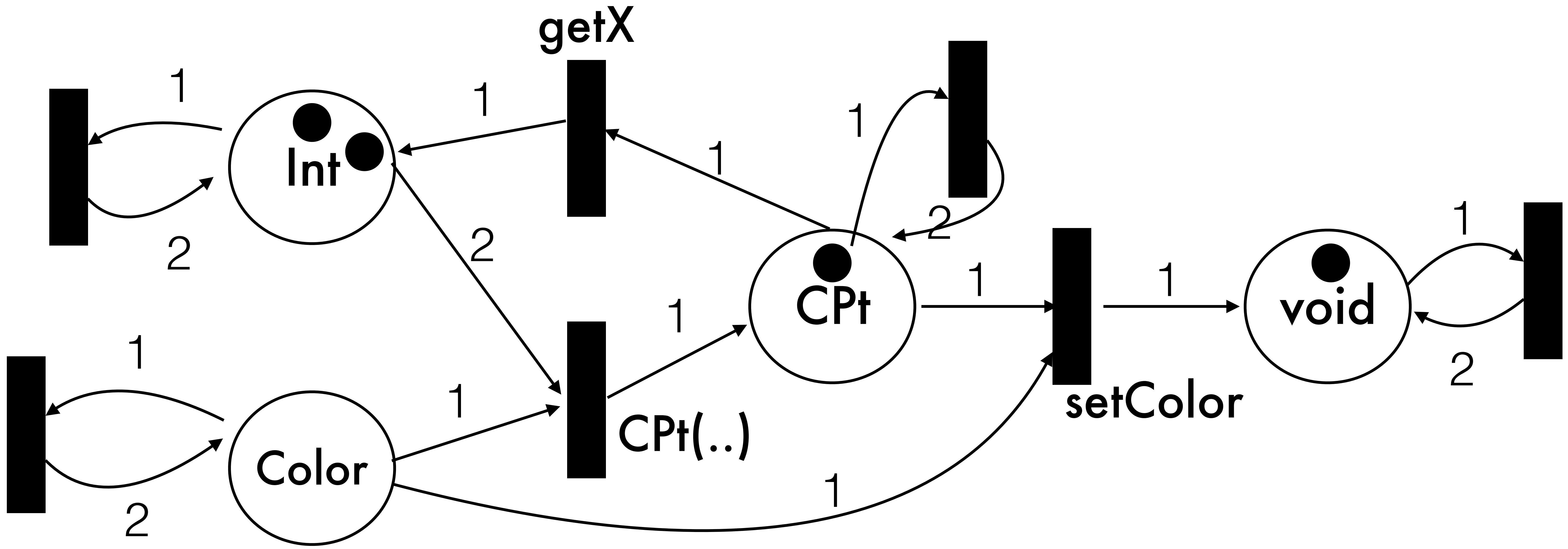


Initial and target markings

Use signature to determine initial and target markings of Petri net

```
CPt shift (CPt p, Int shiftX, Int shiftY)
```

Target marking:



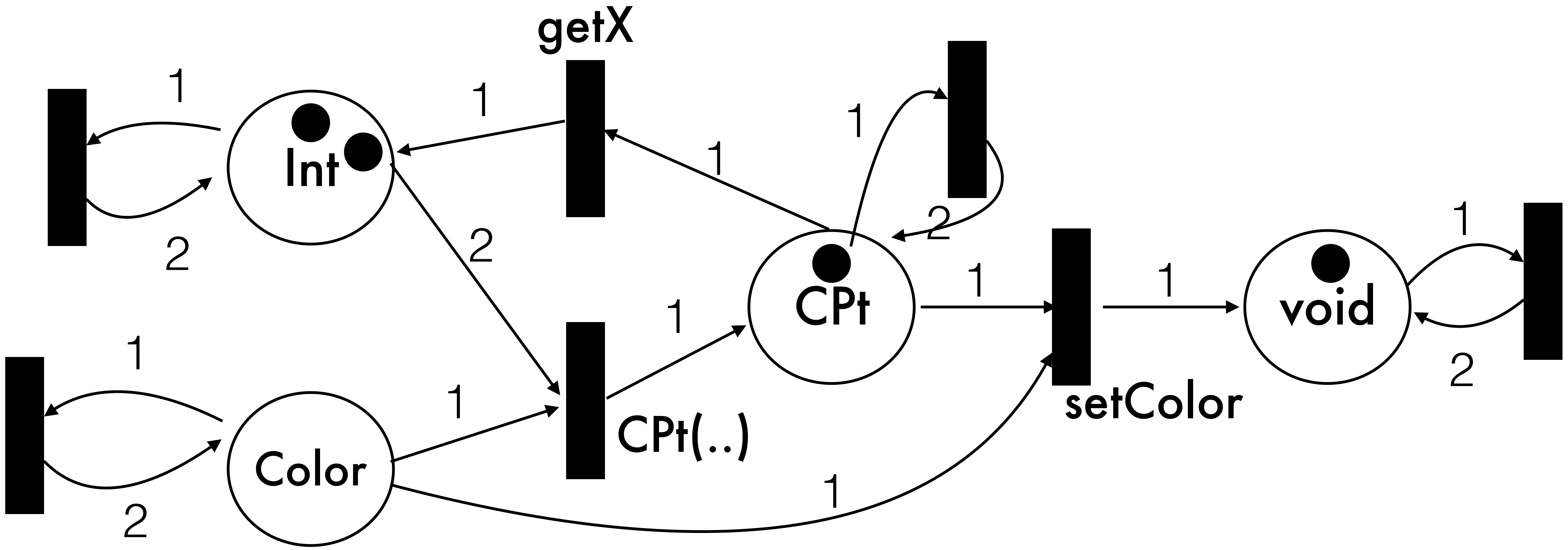
Initial and target markings

Use signature to determine initial and target markings of Petri net

```
CPt shift (CPt p, Int shiftX, Int shiftY)
```

Target marking:

Cpt = 1



Initial and target markings

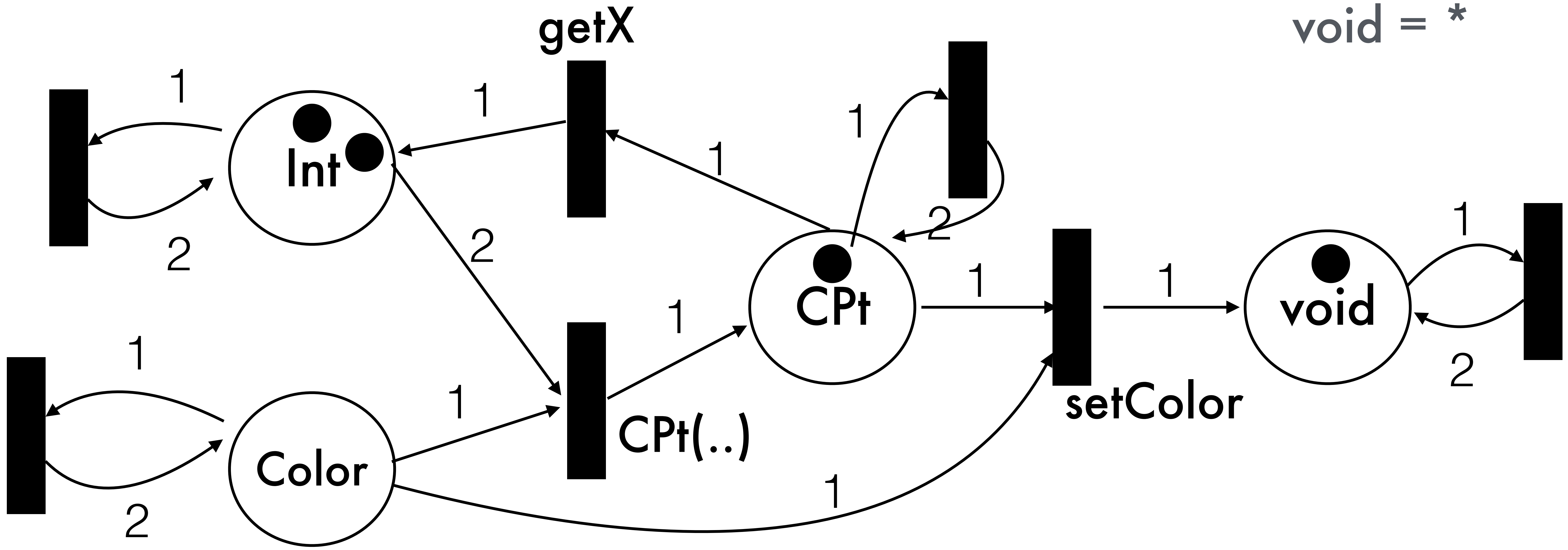
Use signature to determine initial and target markings of Petri net

```
CPt shift (CPt p, Int shiftX, Int shiftY)
```

Target marking:

Cpt = 1

void = *



Initial and target markings

Use signature to determine initial and target markings of Petri net

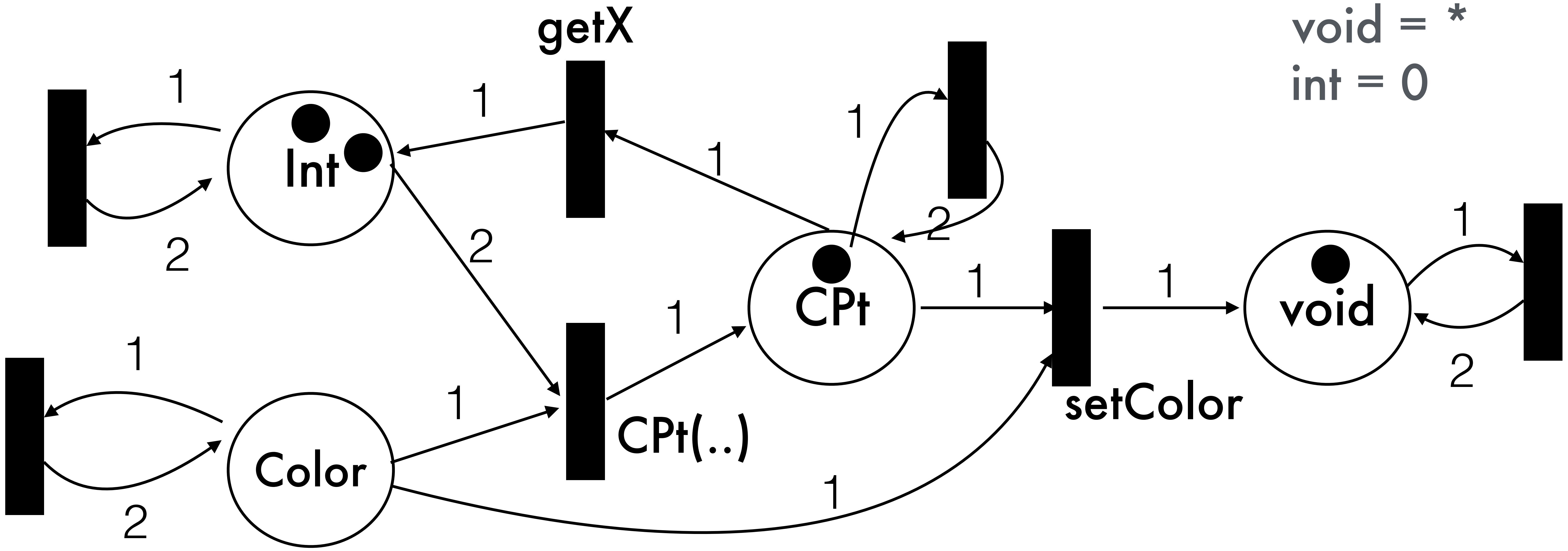
```
CPt shift (CPt p, Int shiftX, Int shiftY)
```

Target marking:

Cpt = 1

void = *

int = 0



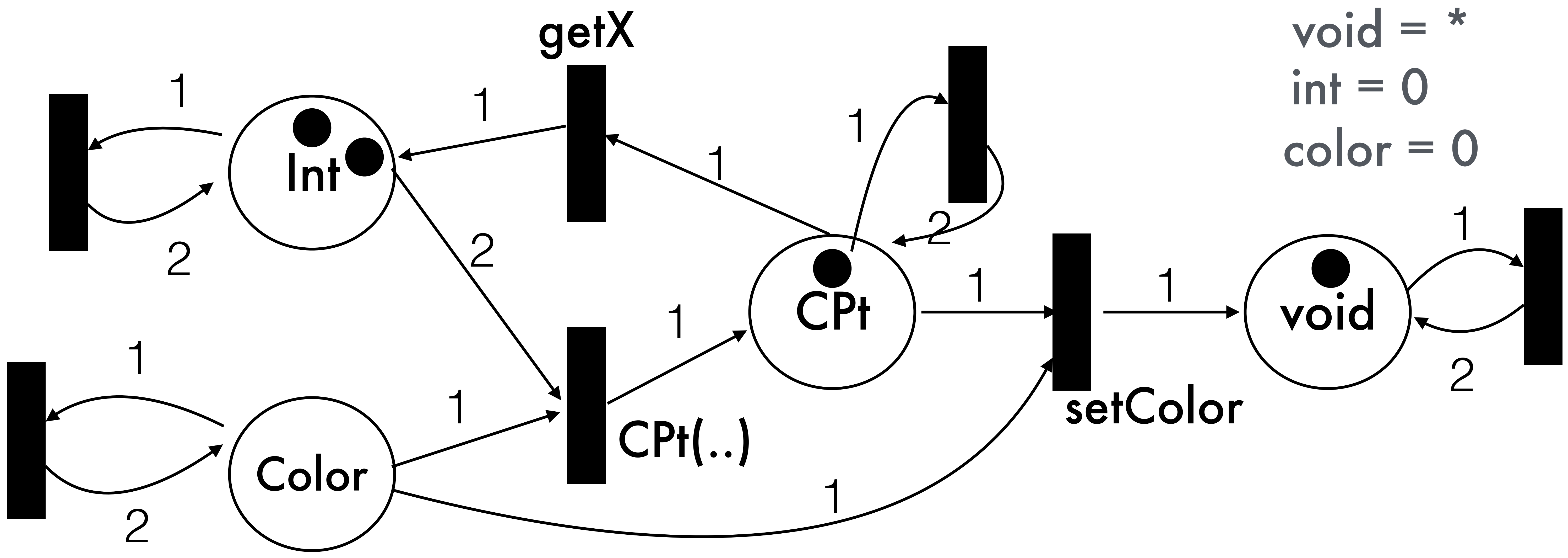
Initial and target markings

Use signature to determine initial and target markings of Petri net

```
CPt shift (CPt p, Int shiftX, Int shiftY)
```

Target marking:

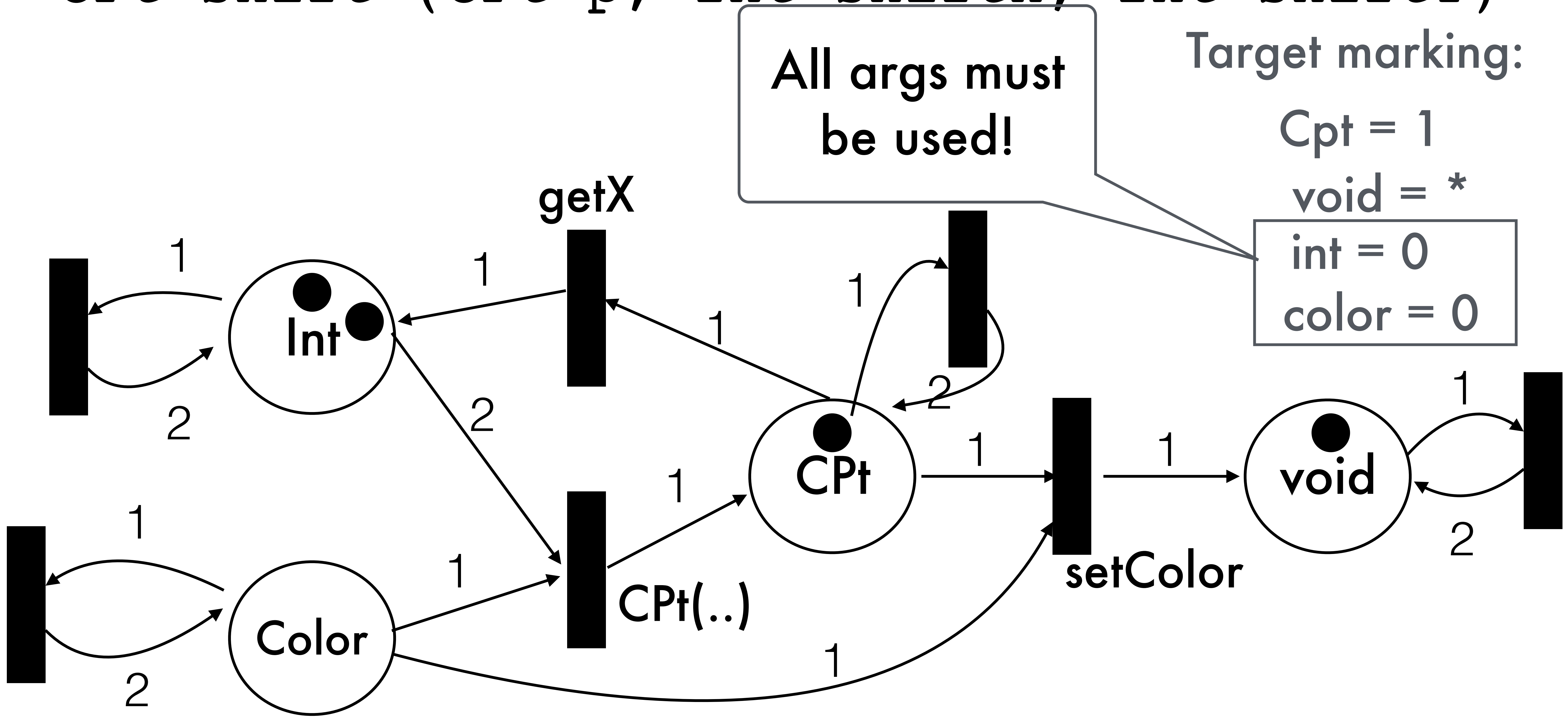
Cpt = 1
void = *
int = 0
color = 0



Initial and target markings

Use signature to determine initial and target markings of Petri net

```
CPt shift (CPt p, Int shiftX, Int shiftY)
```



Exercise 1: Building a Petri Net

```
class Point {
  Point();
  int getX();
  int getY();
  void setX(int);
  void setY(int);
 }
```

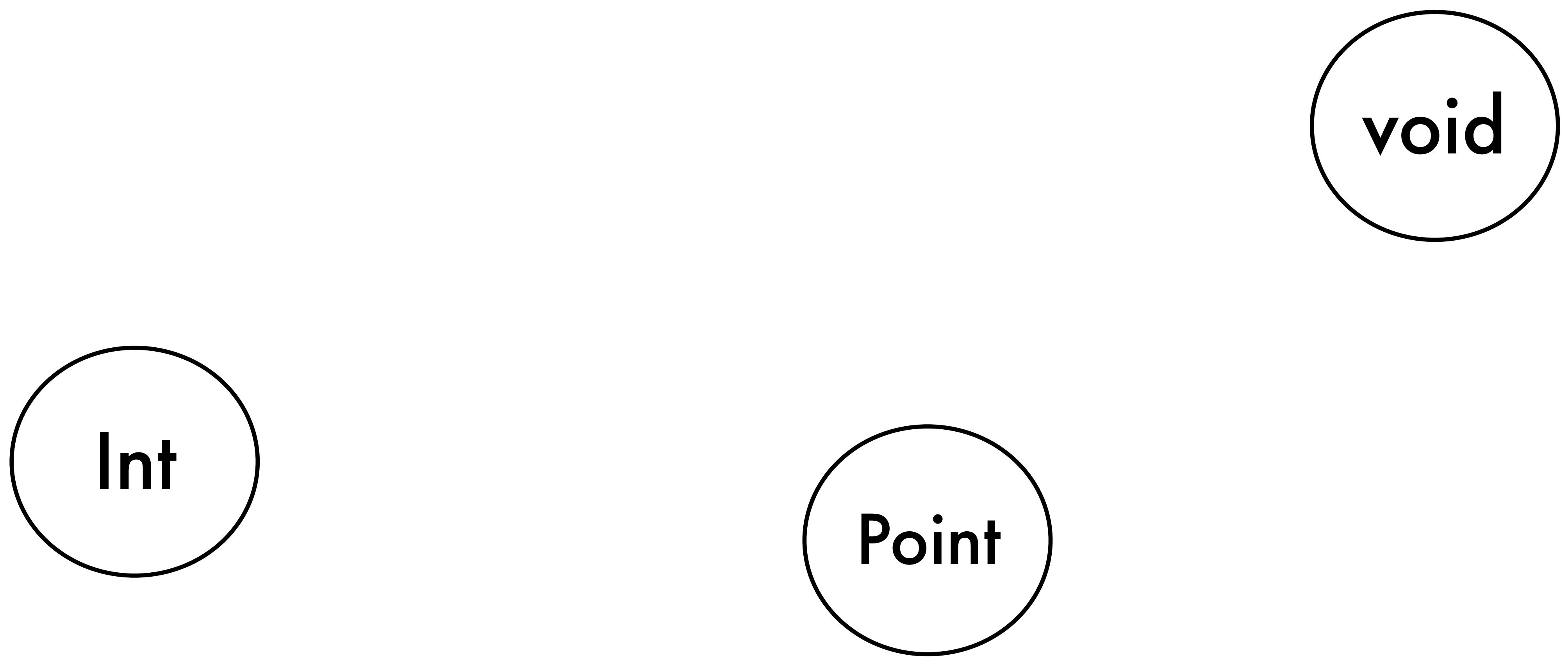
```
class MyPoint {
  MyPoint(int x, int y);
  int getX();
  int getY();
}
```

- Build a petri net with the classes Point and MyPoint:
  - Hint: What are the places (i.e., types)?
  - Hint: What are the transitions (i.e., methods)?
  - Hint: Don't forget the clone edges!

Exercise 1: Solution

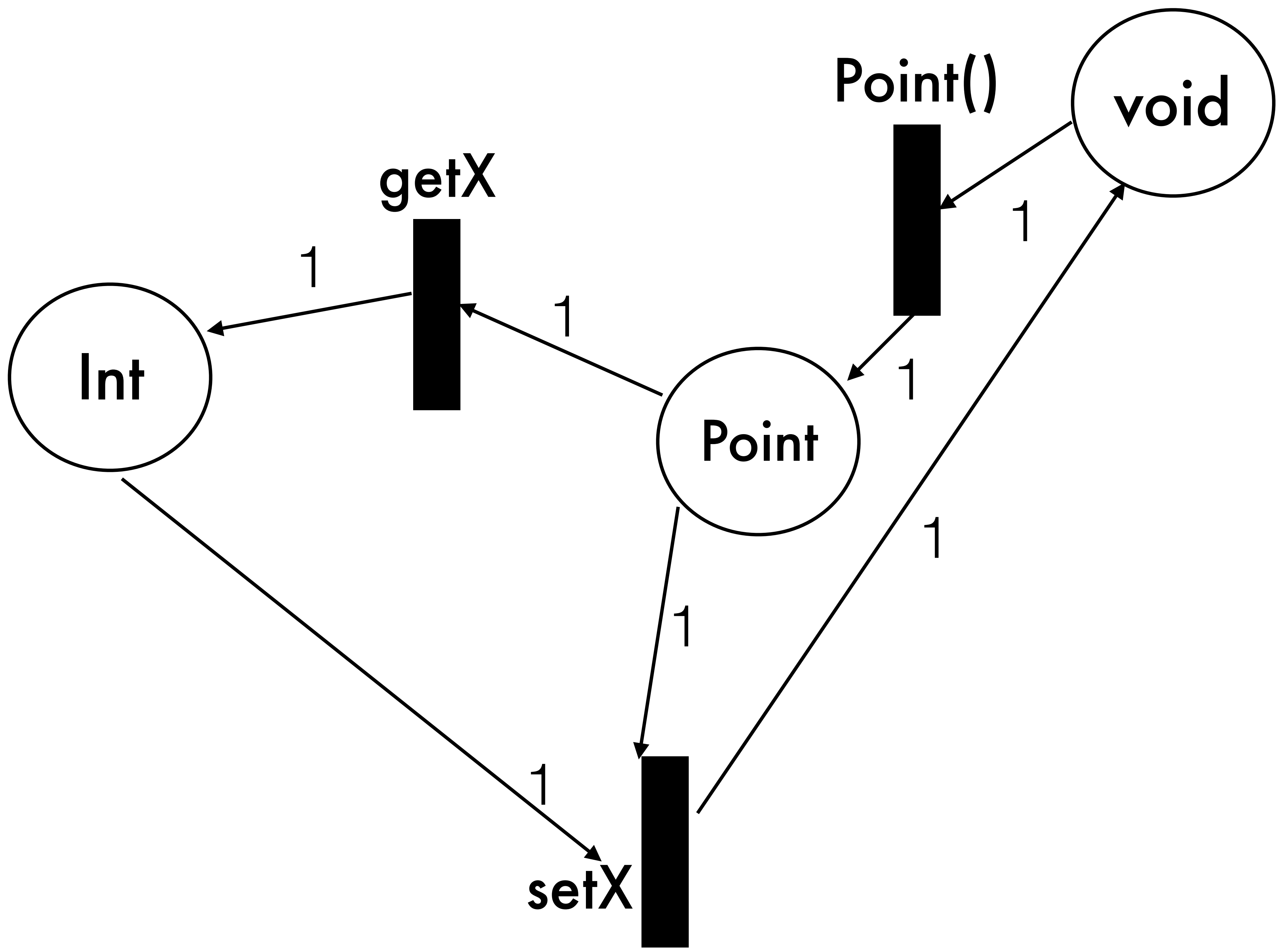
```
class Point {
  Point();
  int getX();
  int getY();
  void setX(int);
  void setY(int);
}
```

Exercise 1: Solution



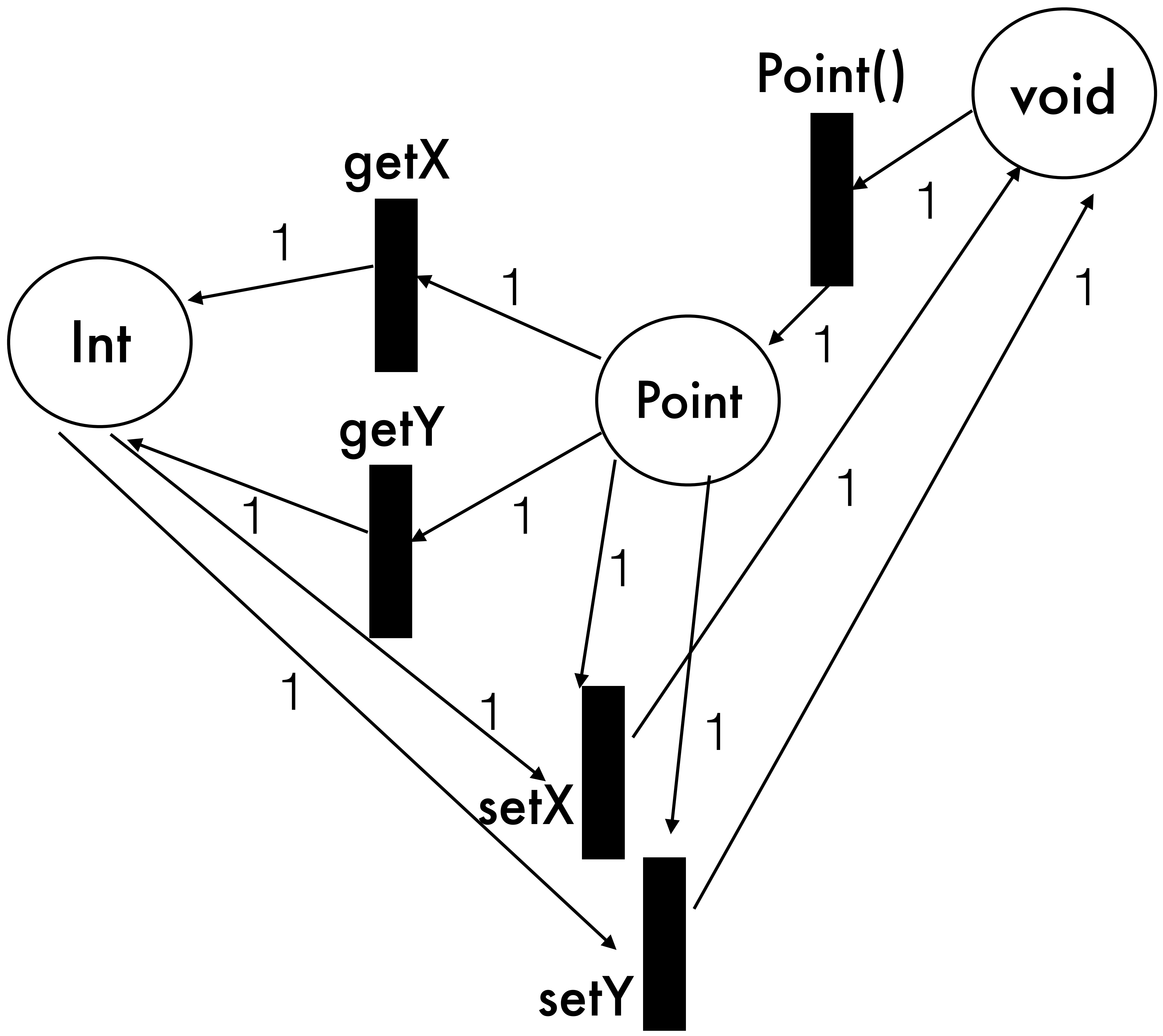

```
class Point {
  Point();
  int getX();
  int getY();
  void setX(int);
  void setY(int);
 }
```

Exercise 1: Solution



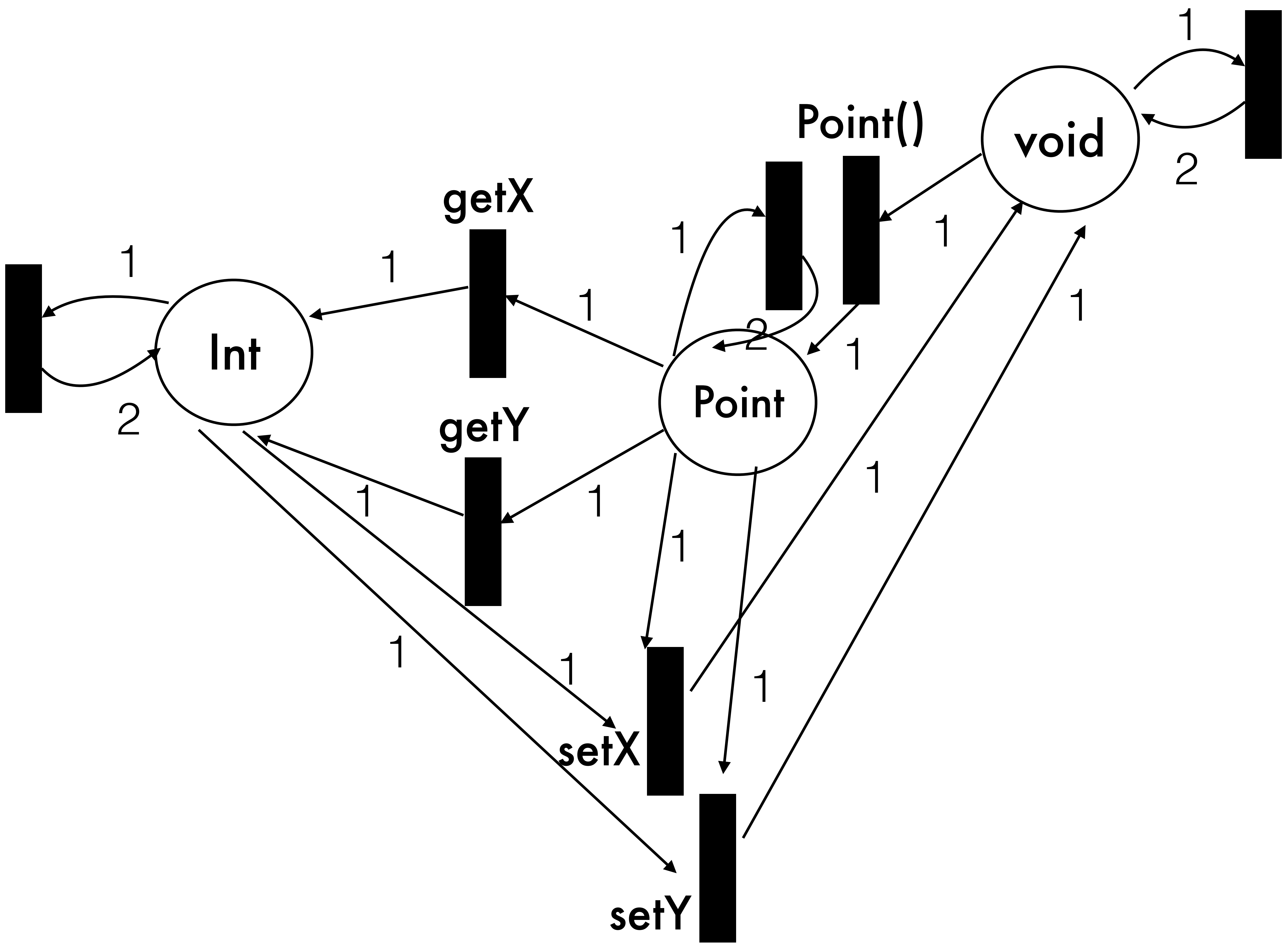

```
class Point {
  Point();
  int getX();
  int getY();
  void setX(int);
  void setY(int);
 }
```

Exercise 1: Solution




```
class Point {
  Point();
  int getX();
  int getY();
  void setX(int);
  void setY(int);
 }
```

Exercise 1: Solution

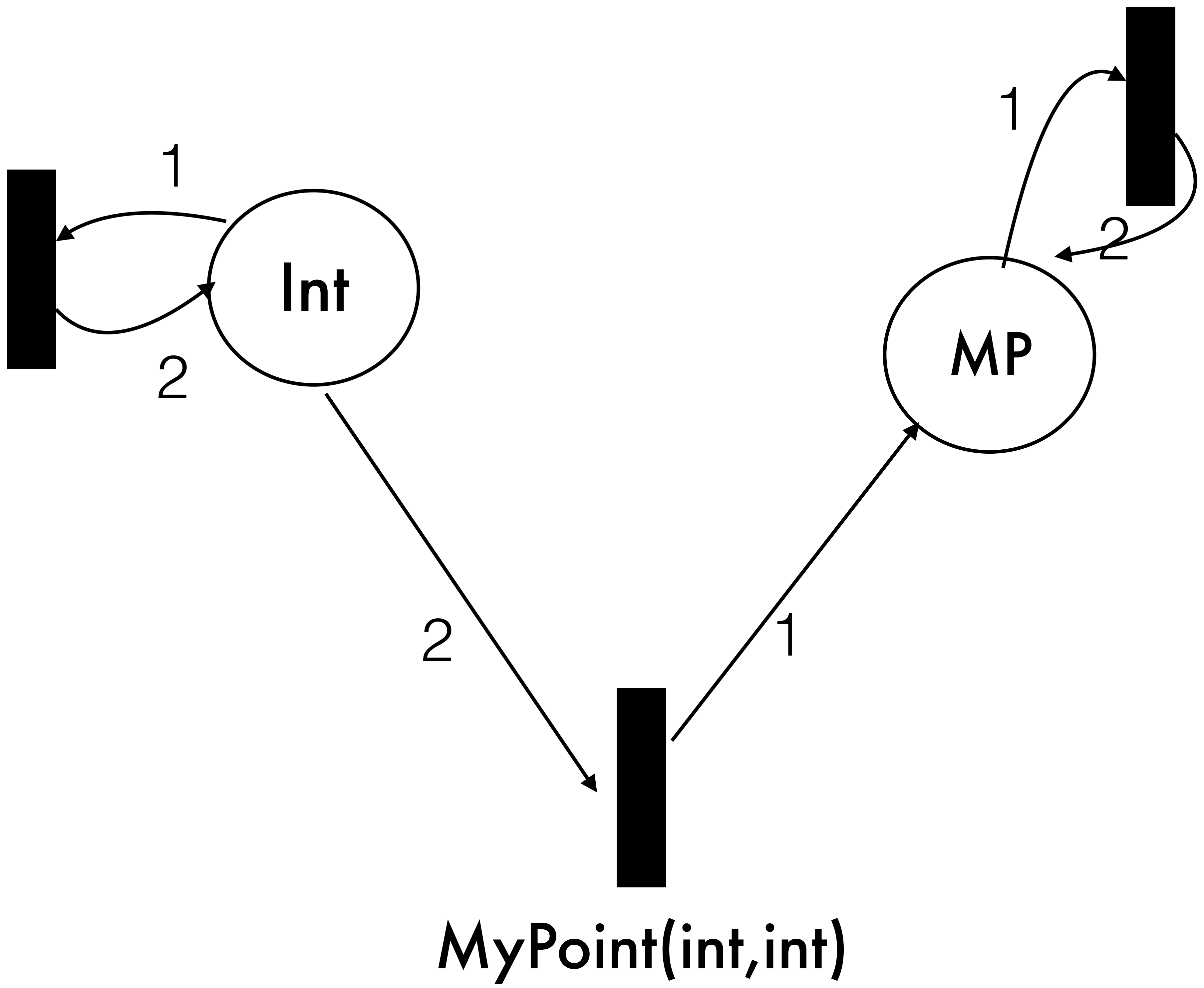



```
class Point {
  Point();
  int getX();
  int getY();
  void setX(int);
  void setY(int);
 }
```

Exercise 1: Solution

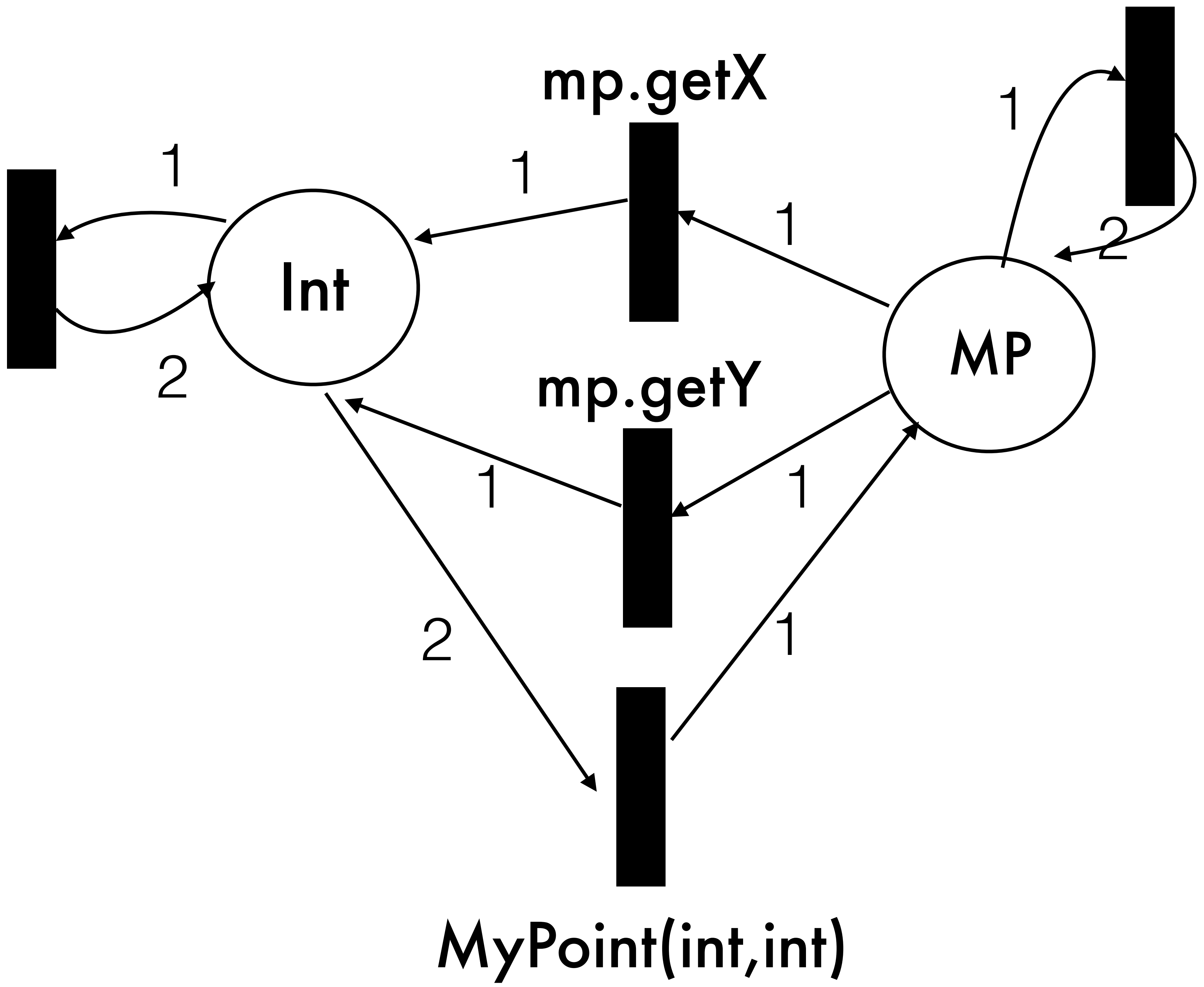
```
class MyPoint {
  MyPoint(int x, int y);
  int getX();
  int getY();
}
```

Exercise 1: Solution



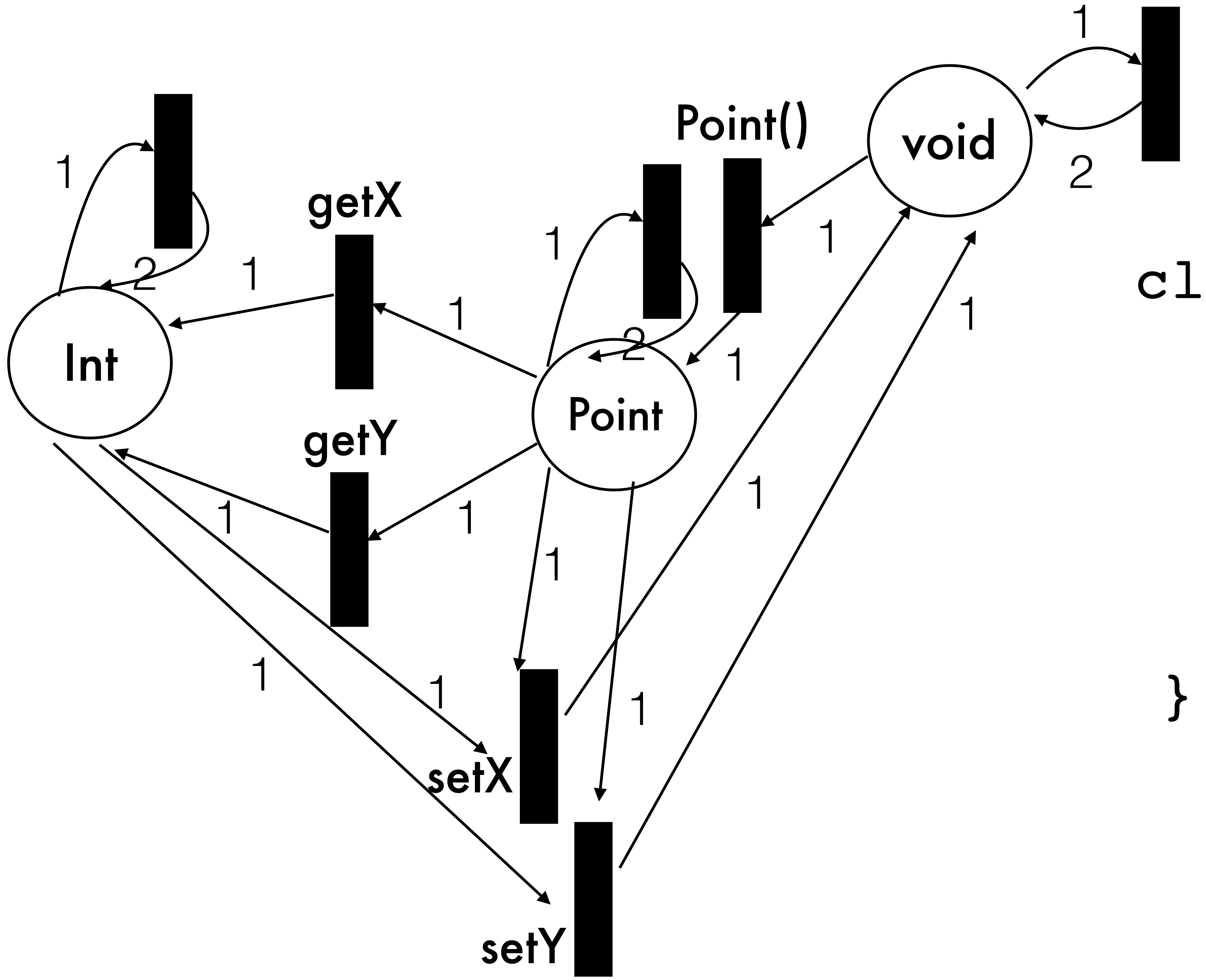

```
class MyPoint {
  MyPoint(int x, int y);
  int getX();
  int getY();
}
```

Exercise 1: Solution



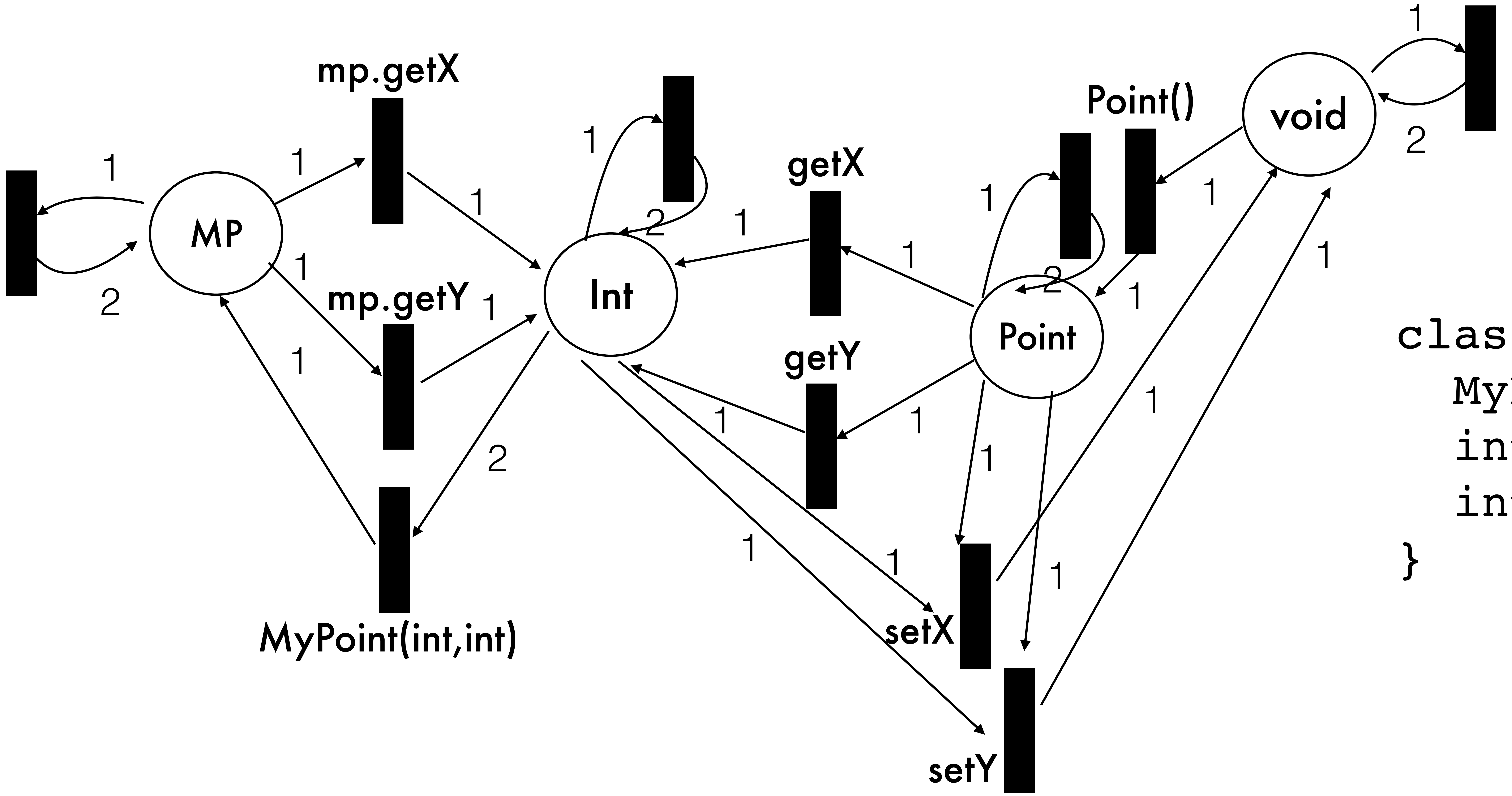

```
class MyPoint {
  MyPoint(int x, int y);
  int getX();
  int getY();
}
```

Exercise 1: Solution



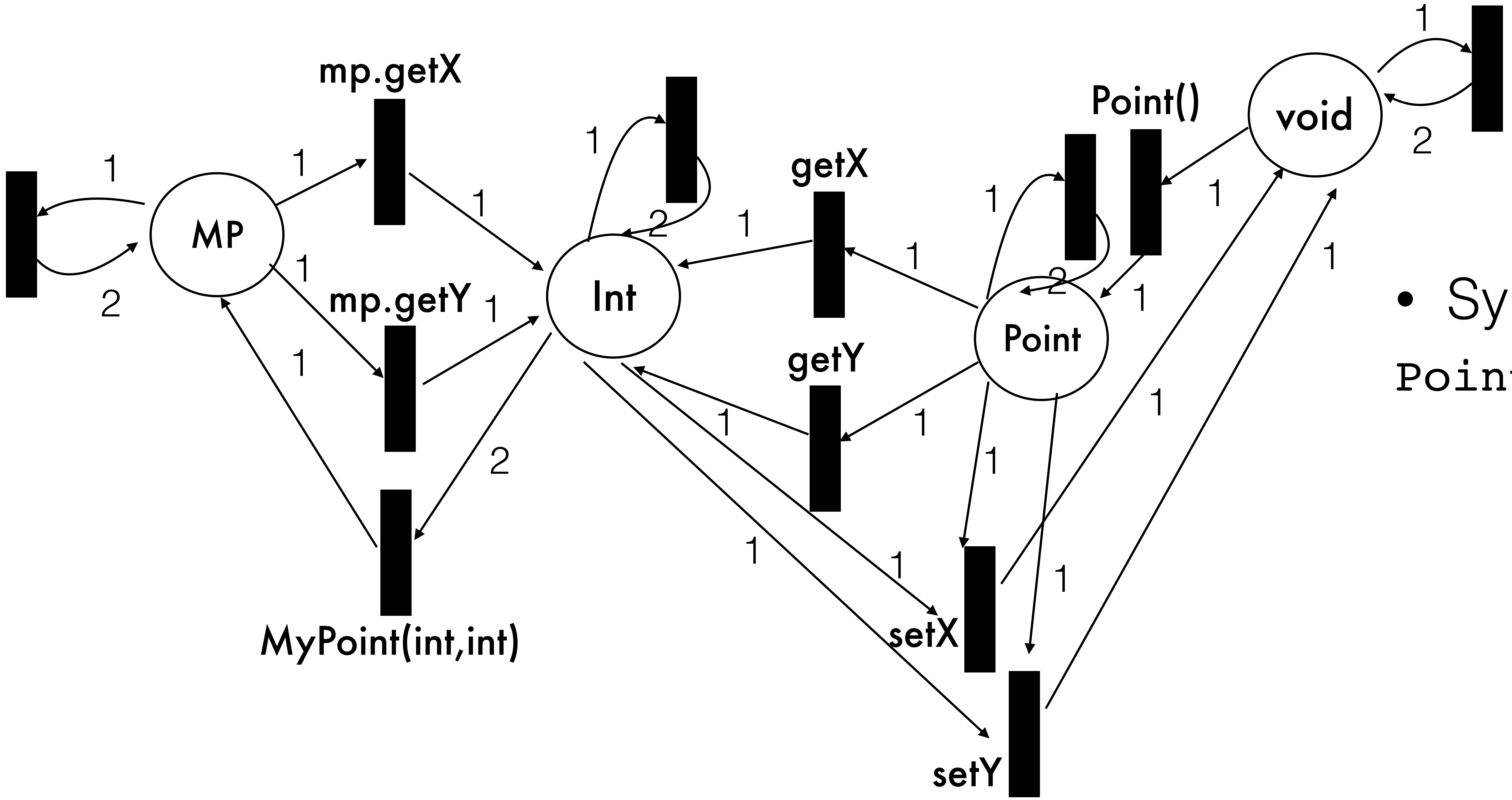

```
class Point {
  Point();
  int getX();
  int getY();
  void setX(int);
  void setY(int);
 }
```

Exercise 1: Solution




```
class MyPoint {
  MyPoint(int x, int y);
  int getX();
  int getY();
}
```

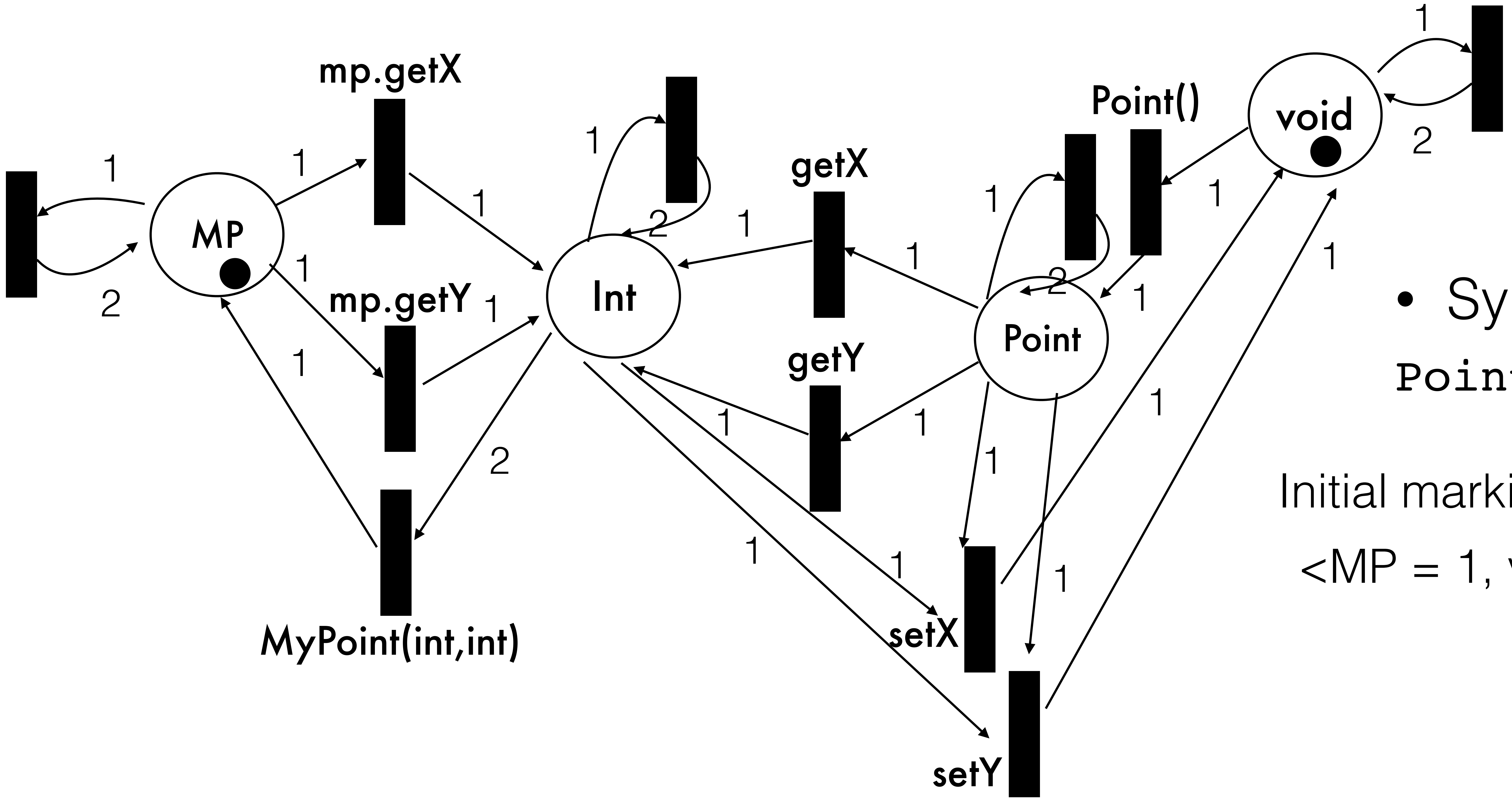
What is the initial marking?



- Synthesize this function:

```
Point convert(Mypoint pt)
```

What is the initial marking?



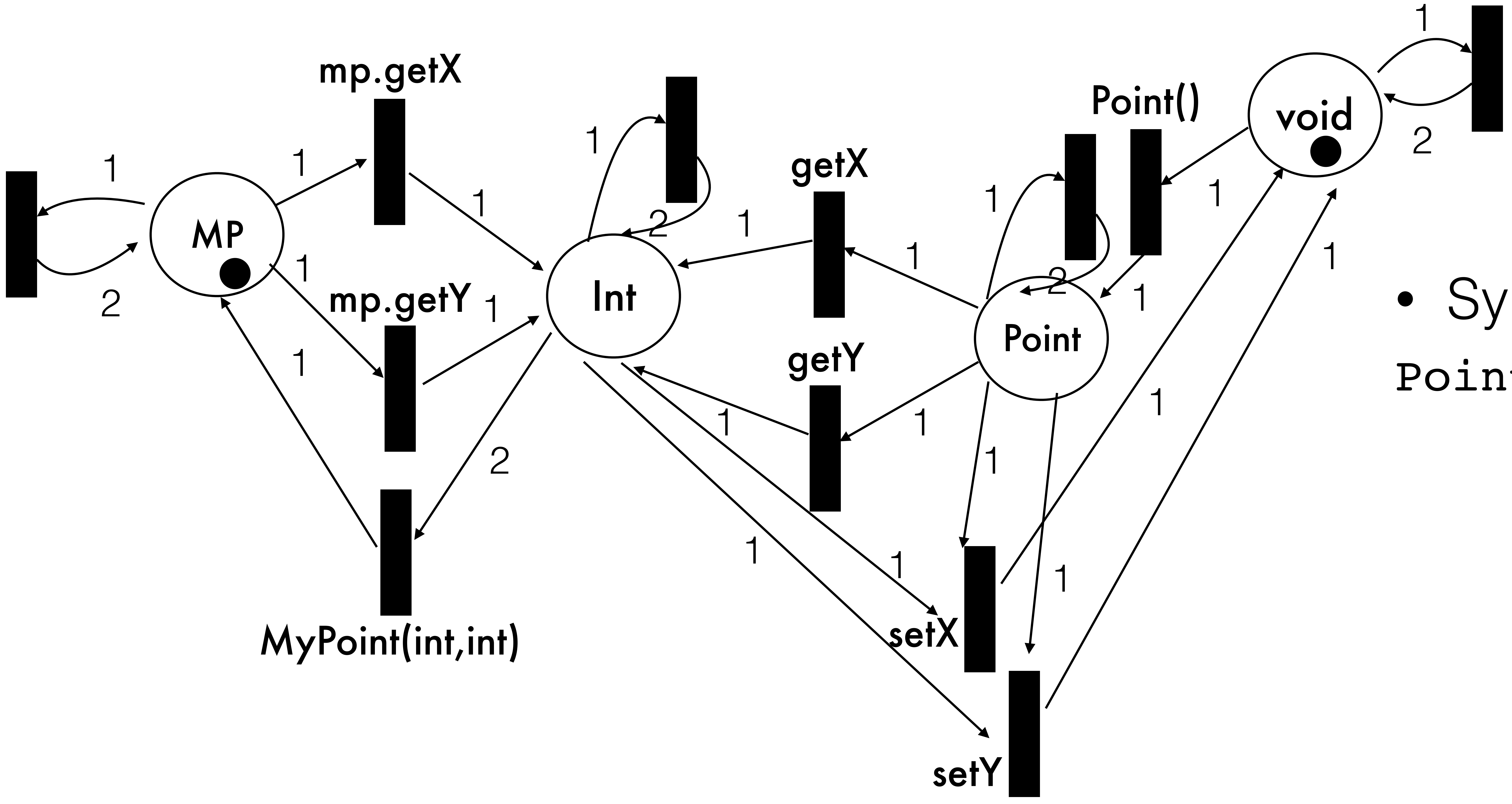
- Synthesize this function:

```
Point convert(Mypoint pt)
```

Initial marking:

<MP = 1, void = 1, Int = 0, Point = 0>

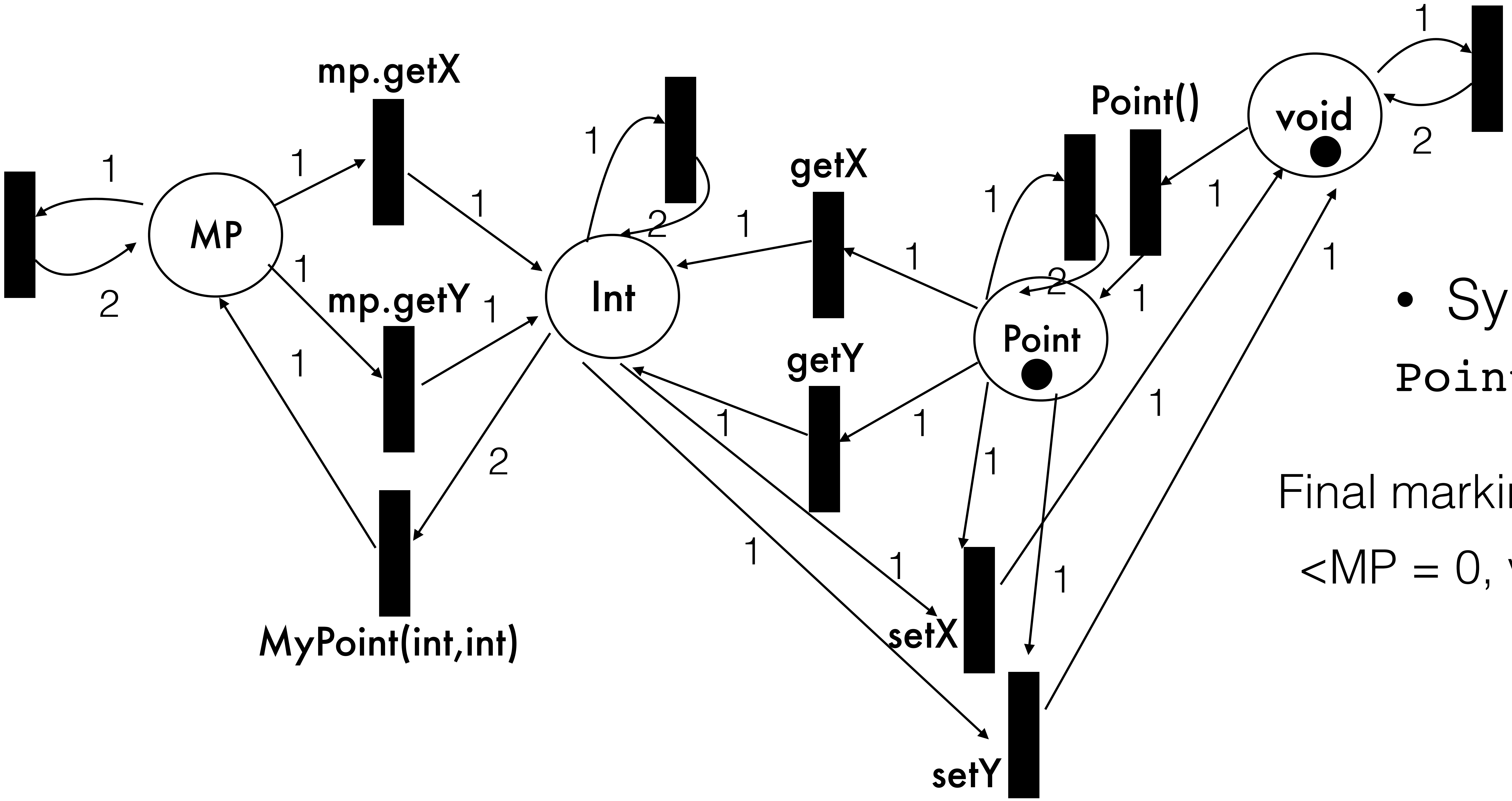
What is the final marking?



- Synthesize this function:

```
Point convert(Mypoint pt)
```

What is the final marking?



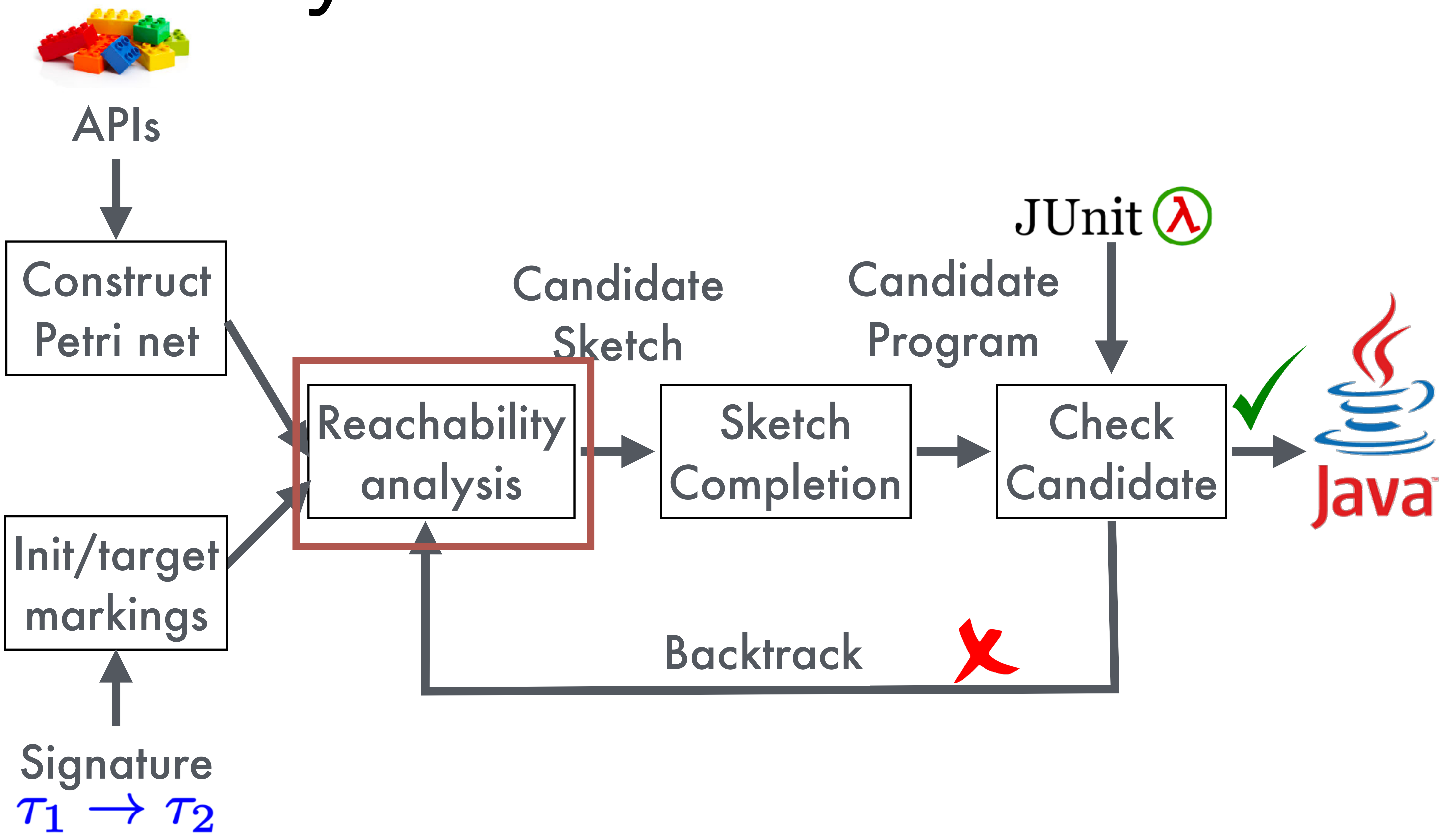
- Synthesize this function:

`Point convert(Mypoint pt)`

Final marking:

<MP = 0, void = *, Int = 0, Point = 1>

SyPet Architecture

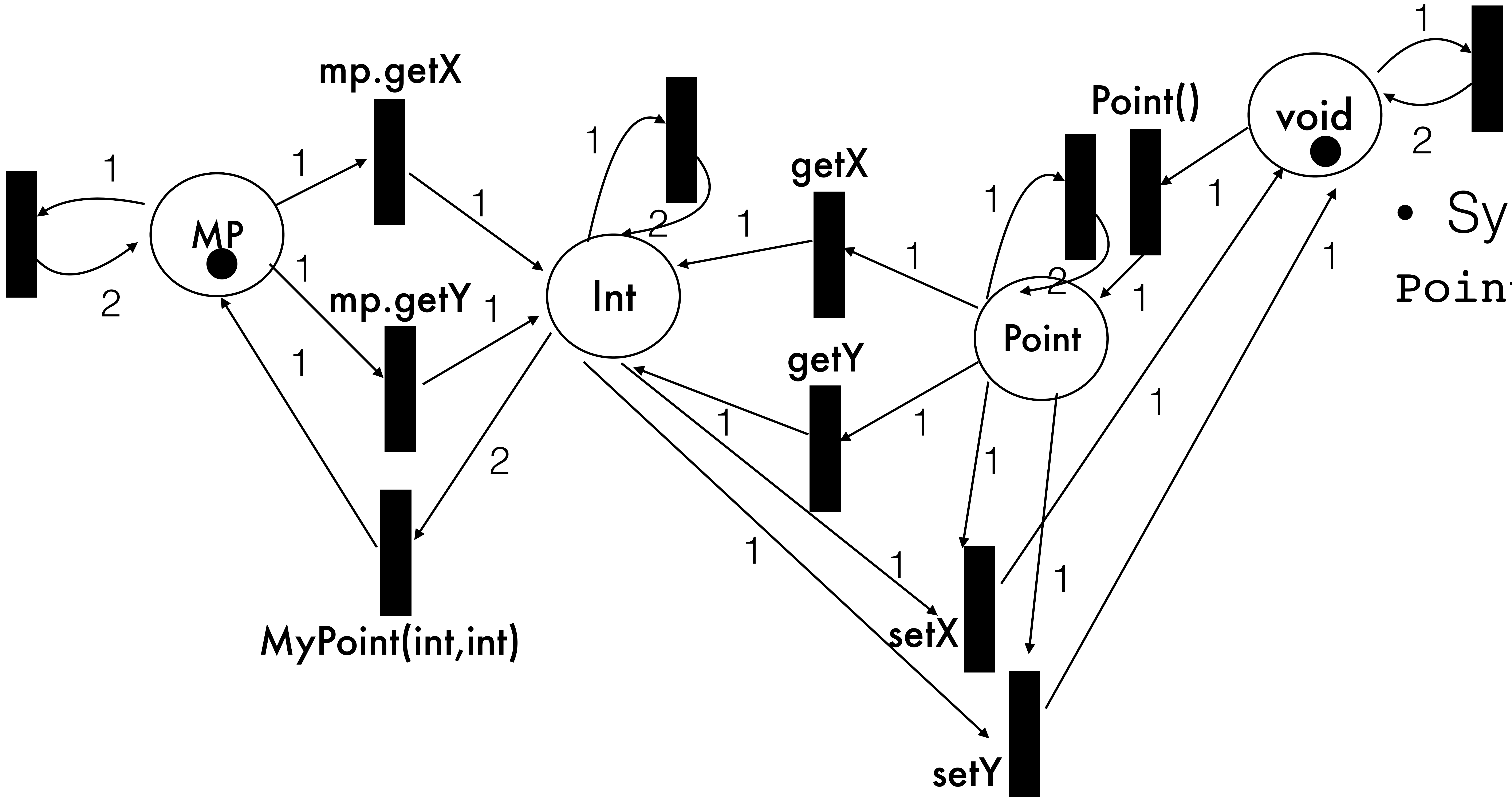


Reachability analysis

All accepting runs of Petri net correspond to method call sequences with desired type signature!

- Need to perform reachability analysis to identify accepting runs of the Petri Net
- Our solution reduces bounded reachability analysis to a SAT problem:
  - Find a reachable path of size *k*
  - Enumerate all reachable paths

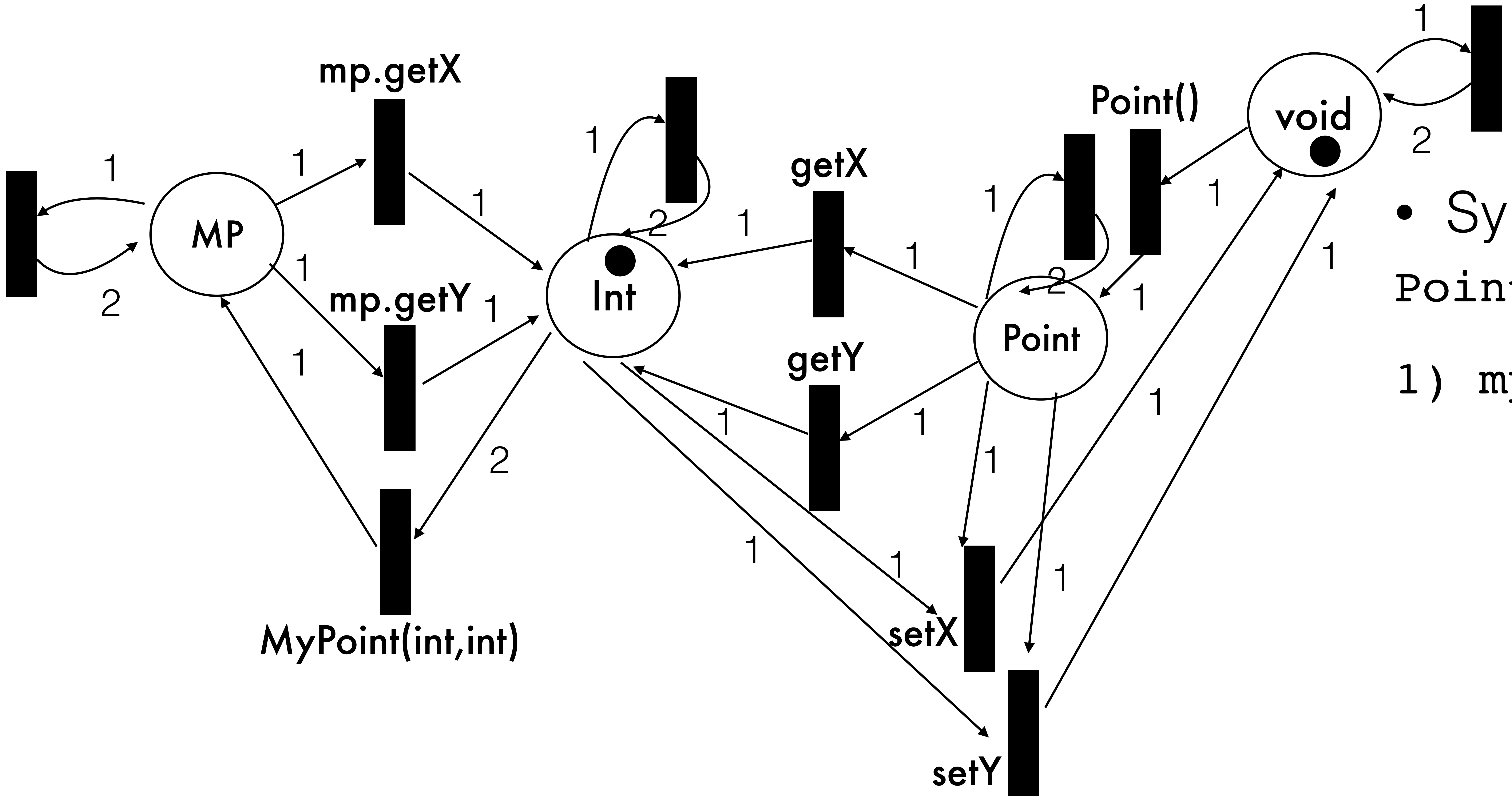
Exercise 2: Reachable paths



- Synthesize this function:

```
Point convert(Mypoint pt)
```

Exercise 2: Reachable paths

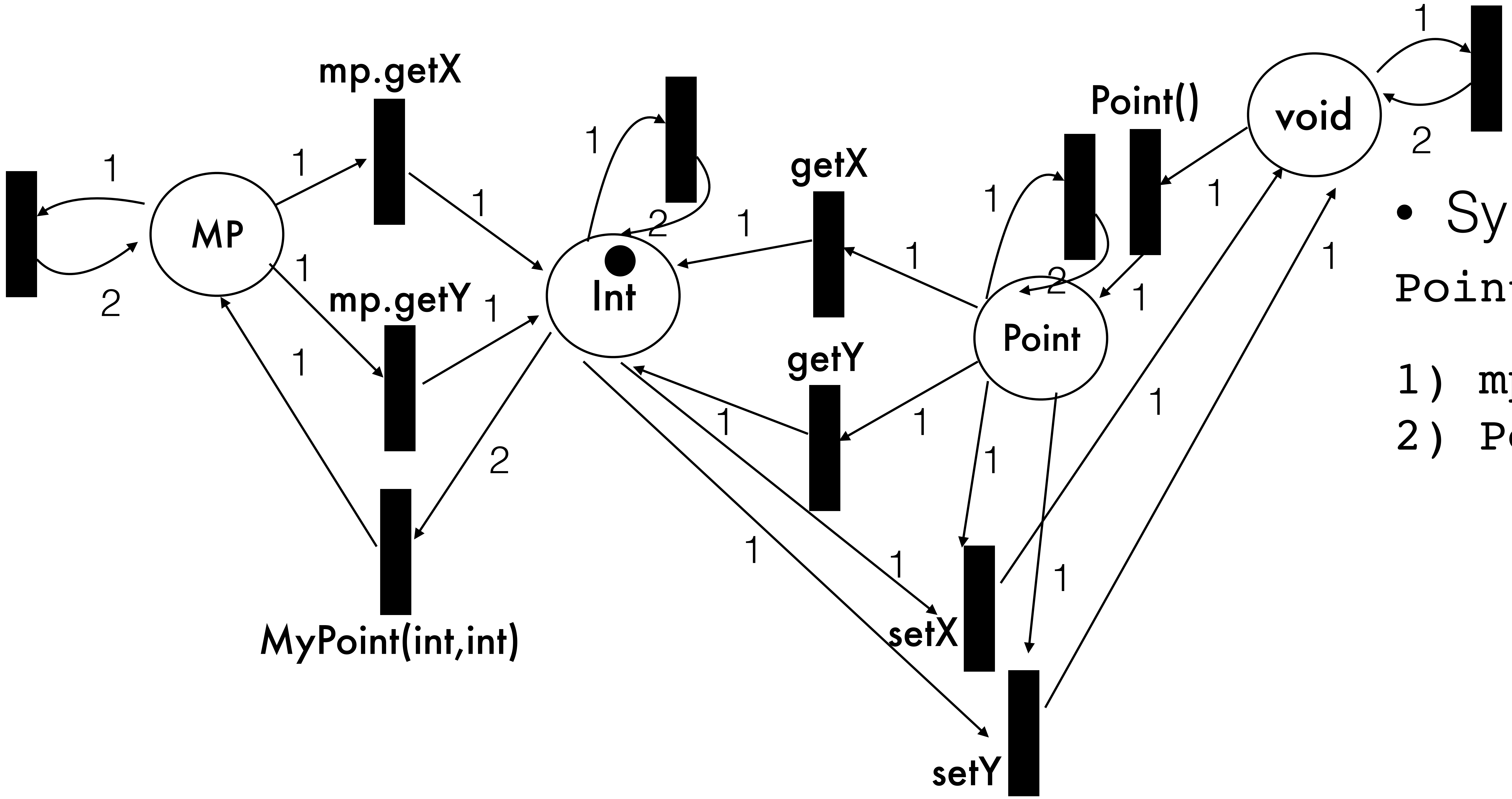


- Synthesize this function:

```
Point convert(Mypoint pt)
```

```
1) mp.getX
```

Exercise 2: Reachable paths

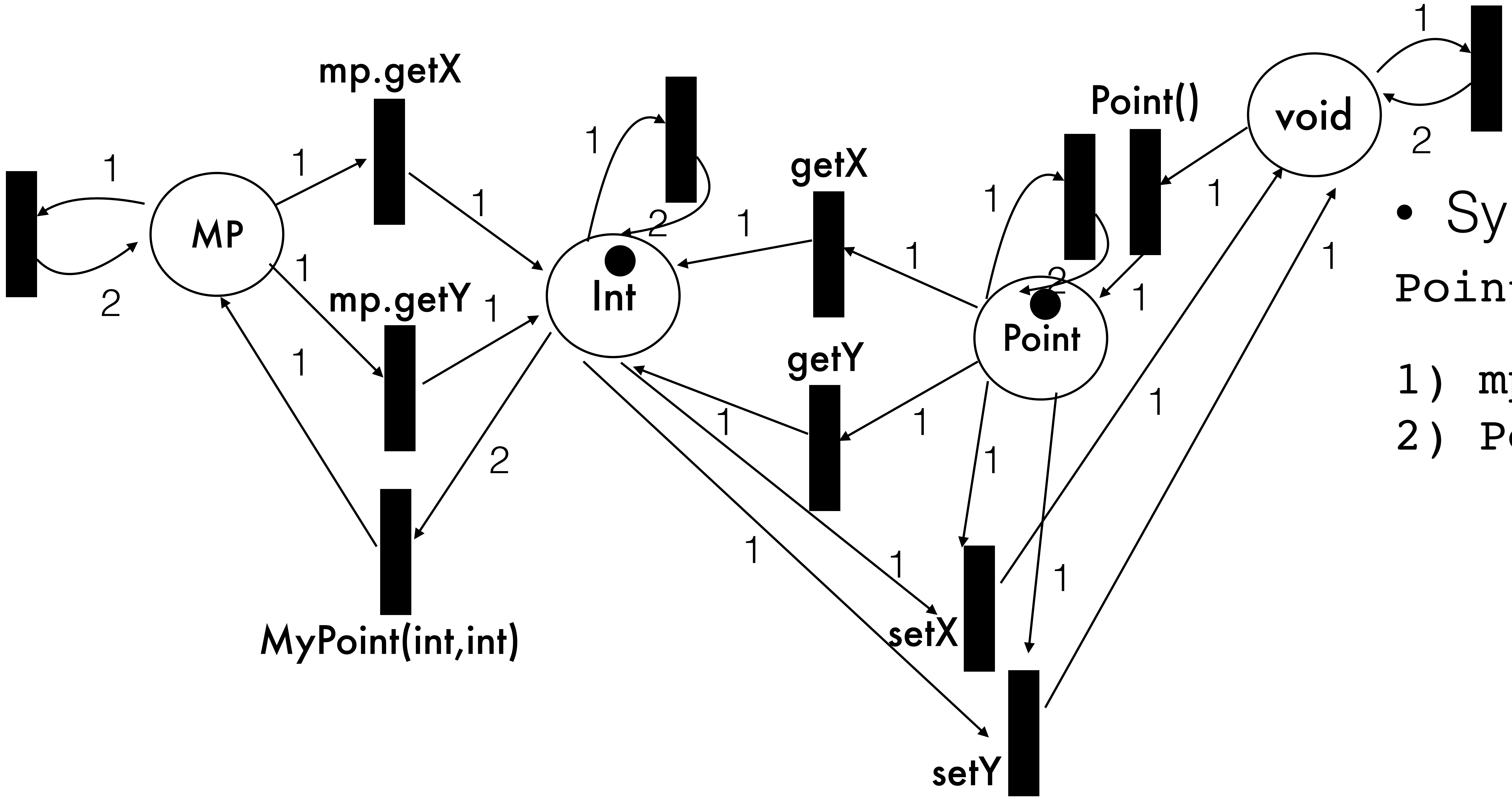


- Synthesize this function:

```
Point convert(Mypoint pt)
```

```
1) mp.getX
2) Point()
```

Exercise 2: Reachable paths

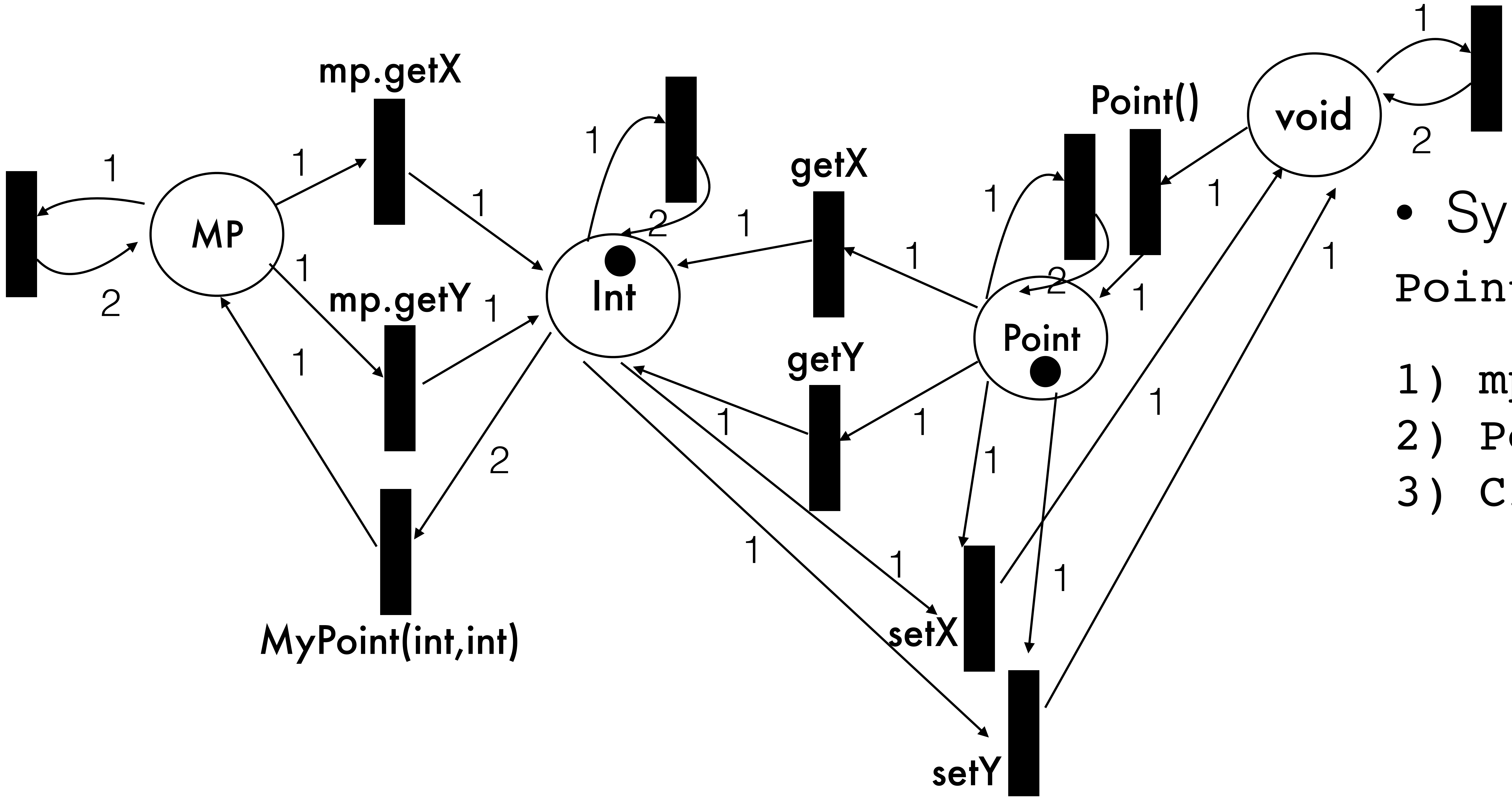


- Synthesize this function:

```
Point convert(Mypoint pt)
```

```
1) mp.getX
2) Point()
```

Exercise 2: Reachable paths

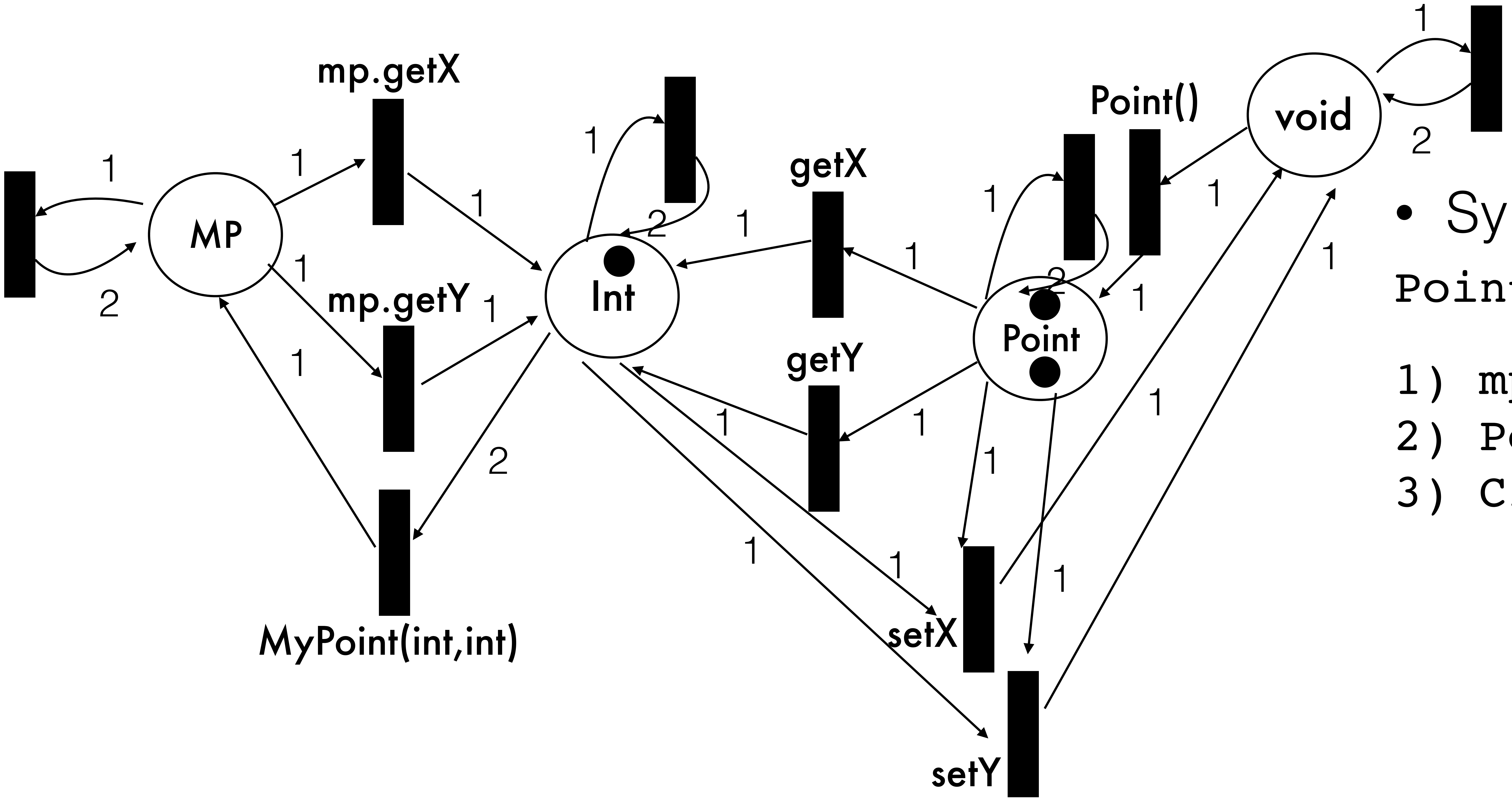


- Synthesize this function:

```
Point convert(Mypoint pt)
```

```
1) mp.getX
2) Point()
3) Clone-Point
```

Exercise 2: Reachable paths

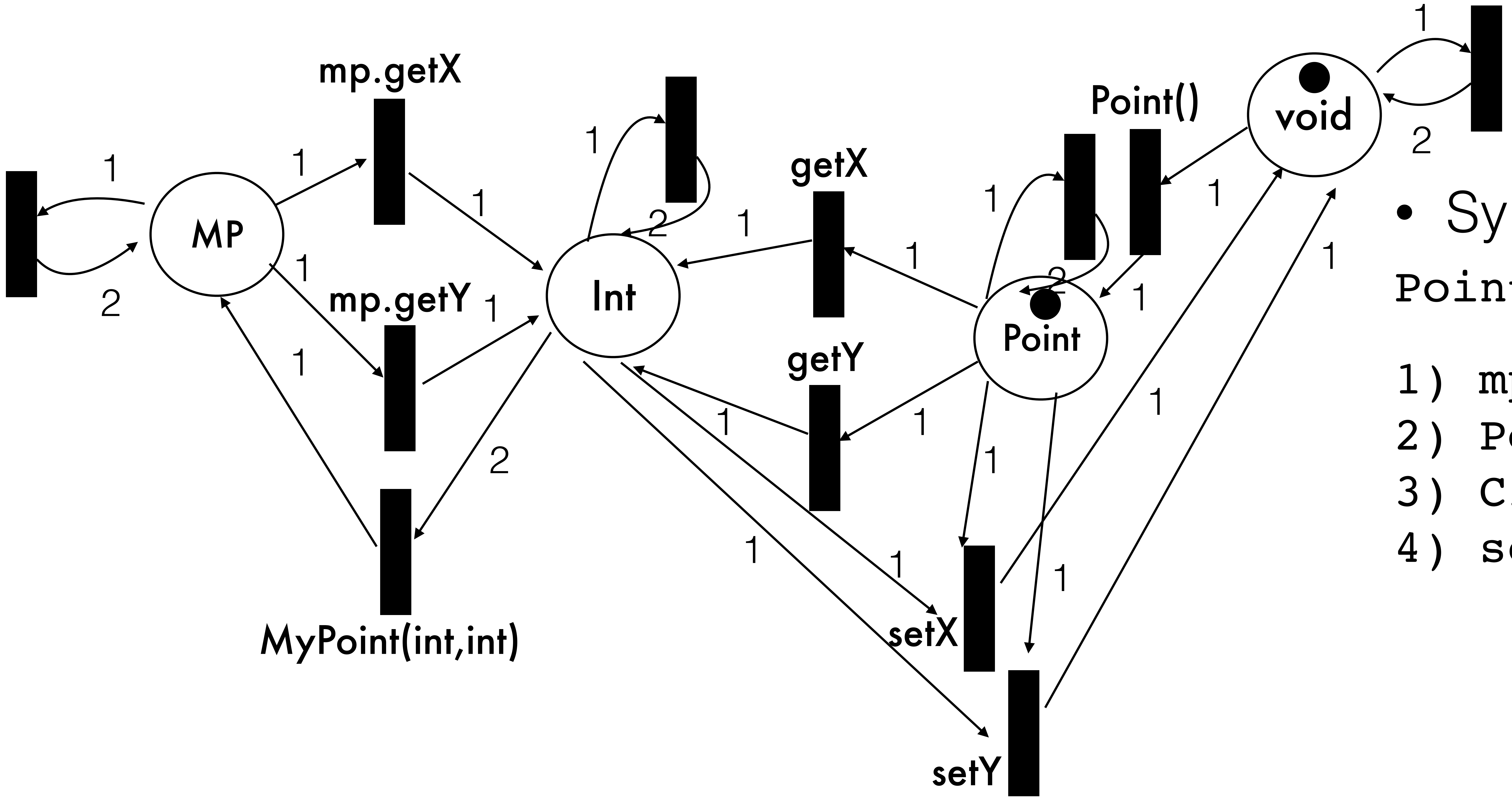


- Synthesize this function:

```
Point convert(Mypoint pt)
```

```
1) mp.getX
2) Point()
3) Clone-Point
```

Exercise 2: Reachable paths

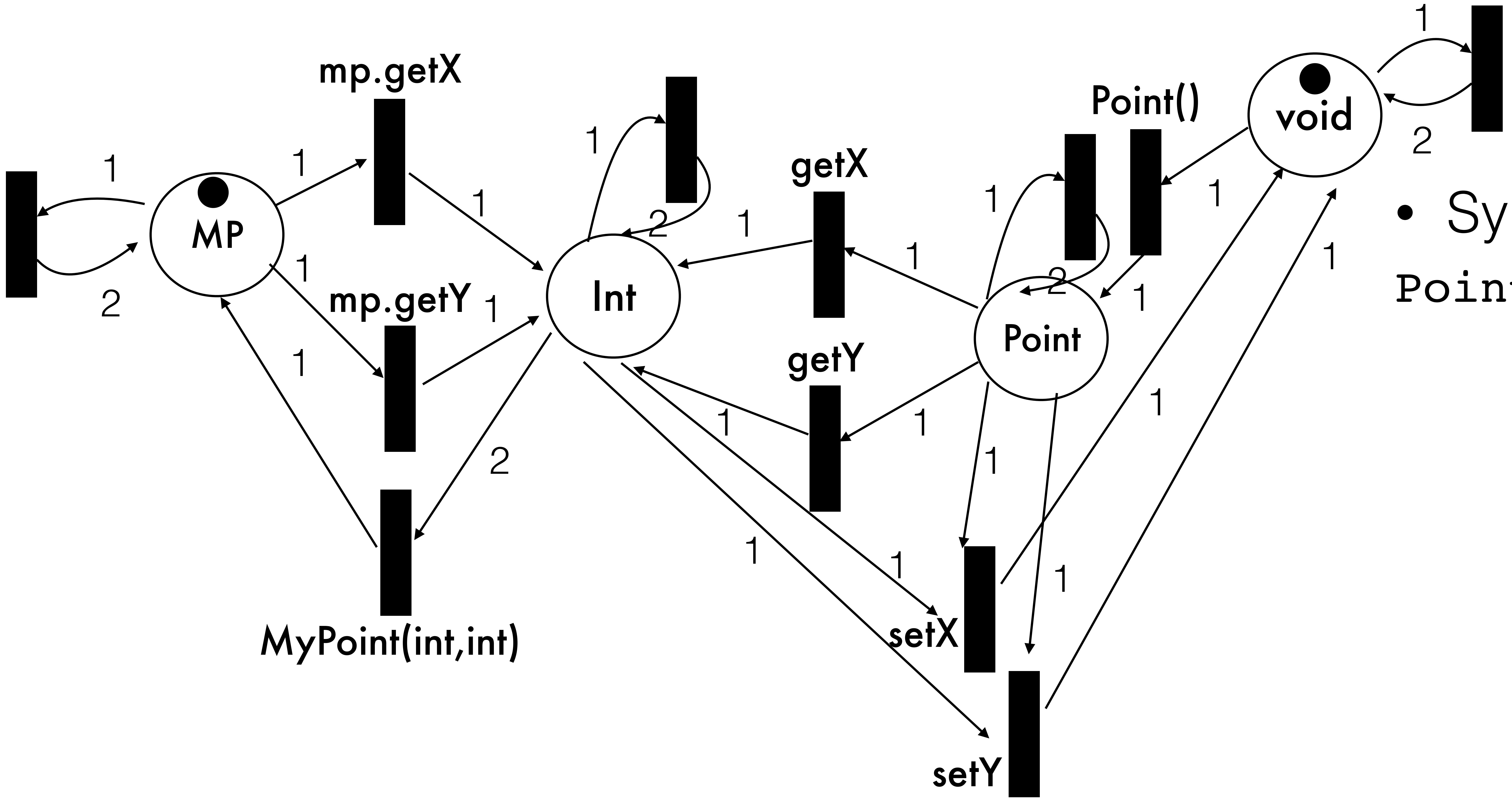


- Synthesize this function:

```
Point convert(Mypoint pt)
```

```
1) mp.getX
2) Point()
3) Clone-Point
4) setX
```

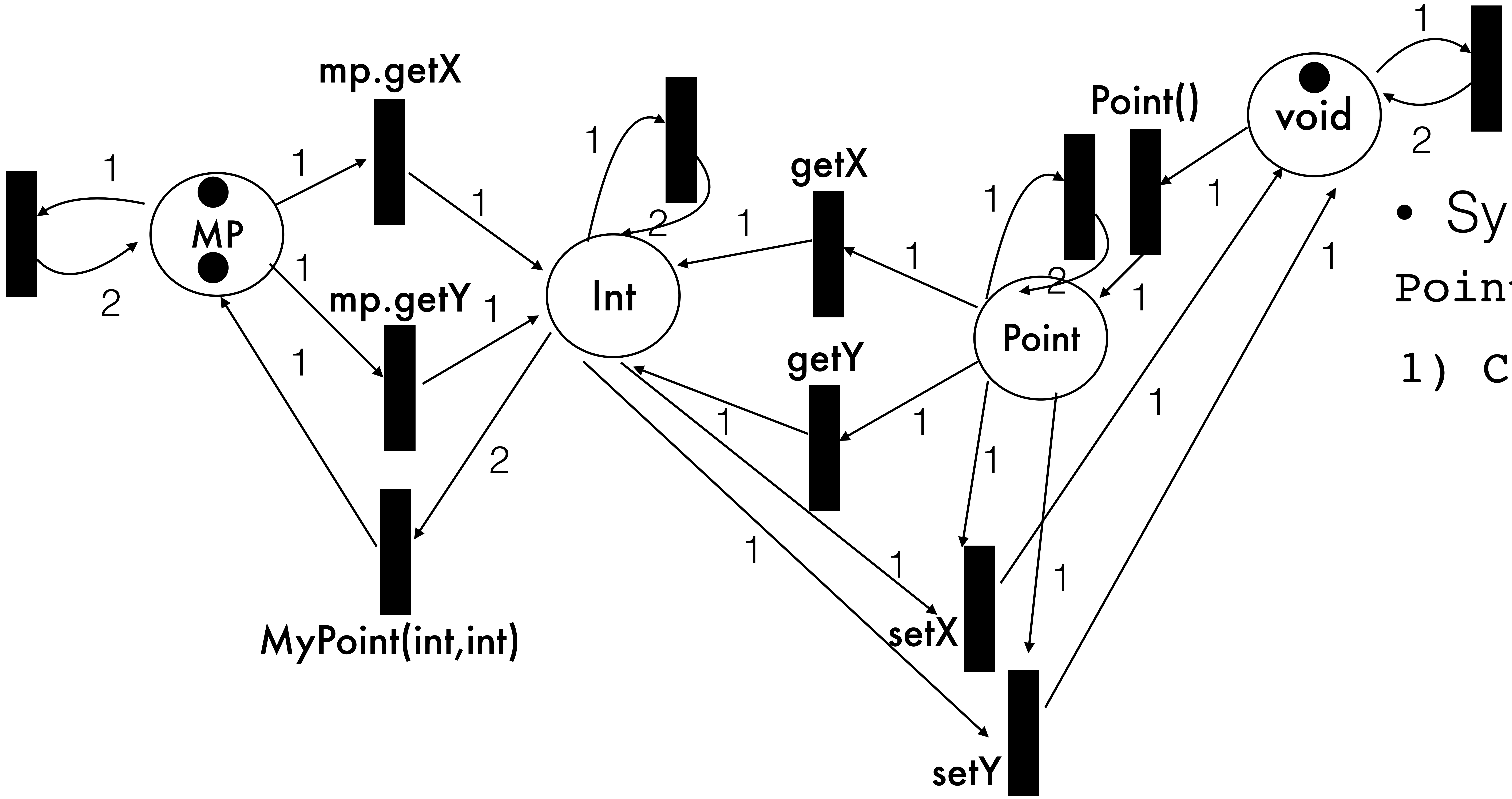
Exercise 2: Reachable paths



- Synthesize this function:

```
Point convert(Mypoint pt)
```

Exercise 2: Reachable paths

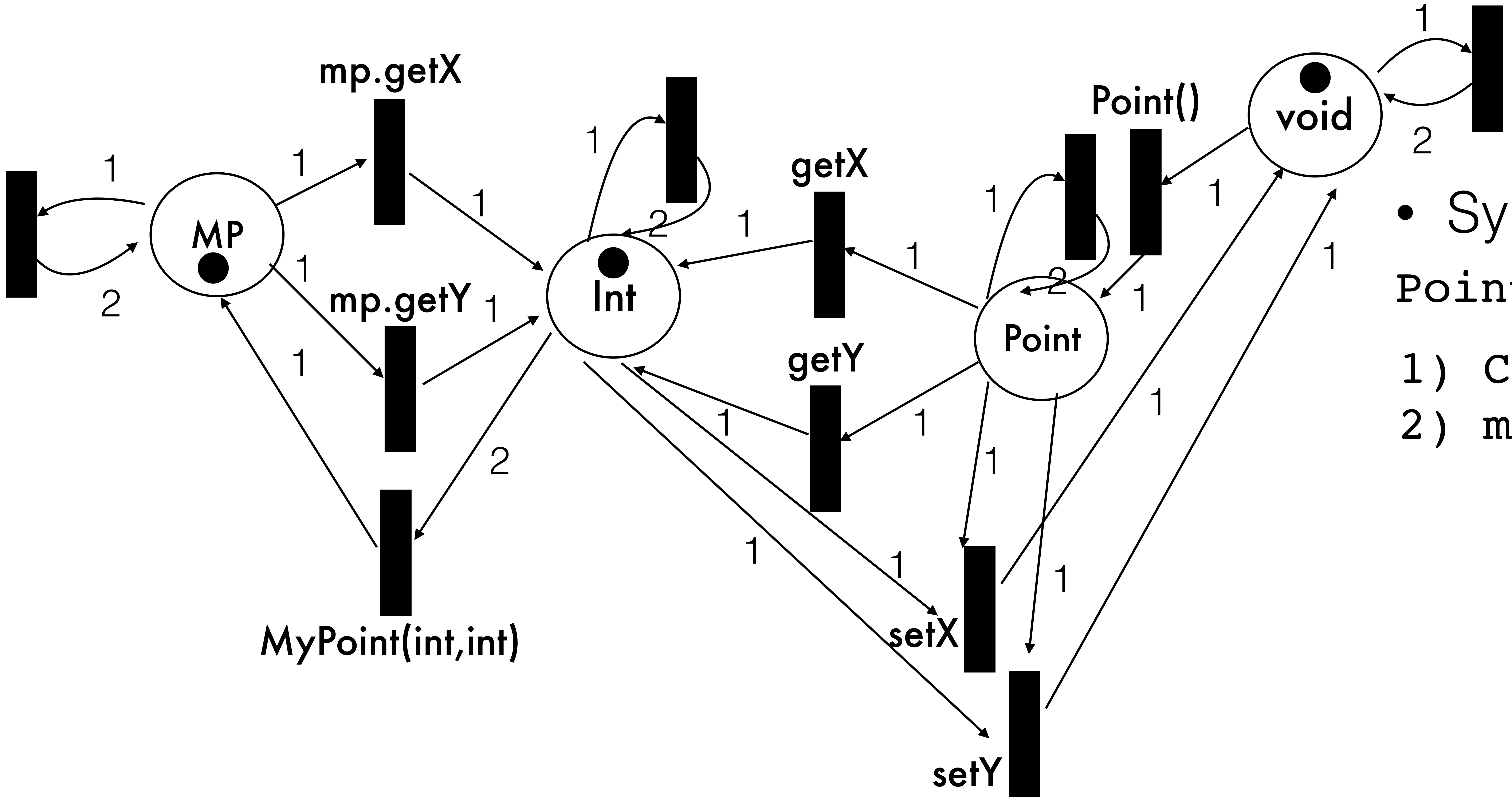


- Synthesize this function:

```
Point convert(Mypoint pt)
```

```
1) Clone-MP
```

Exercise 2: Reachable paths

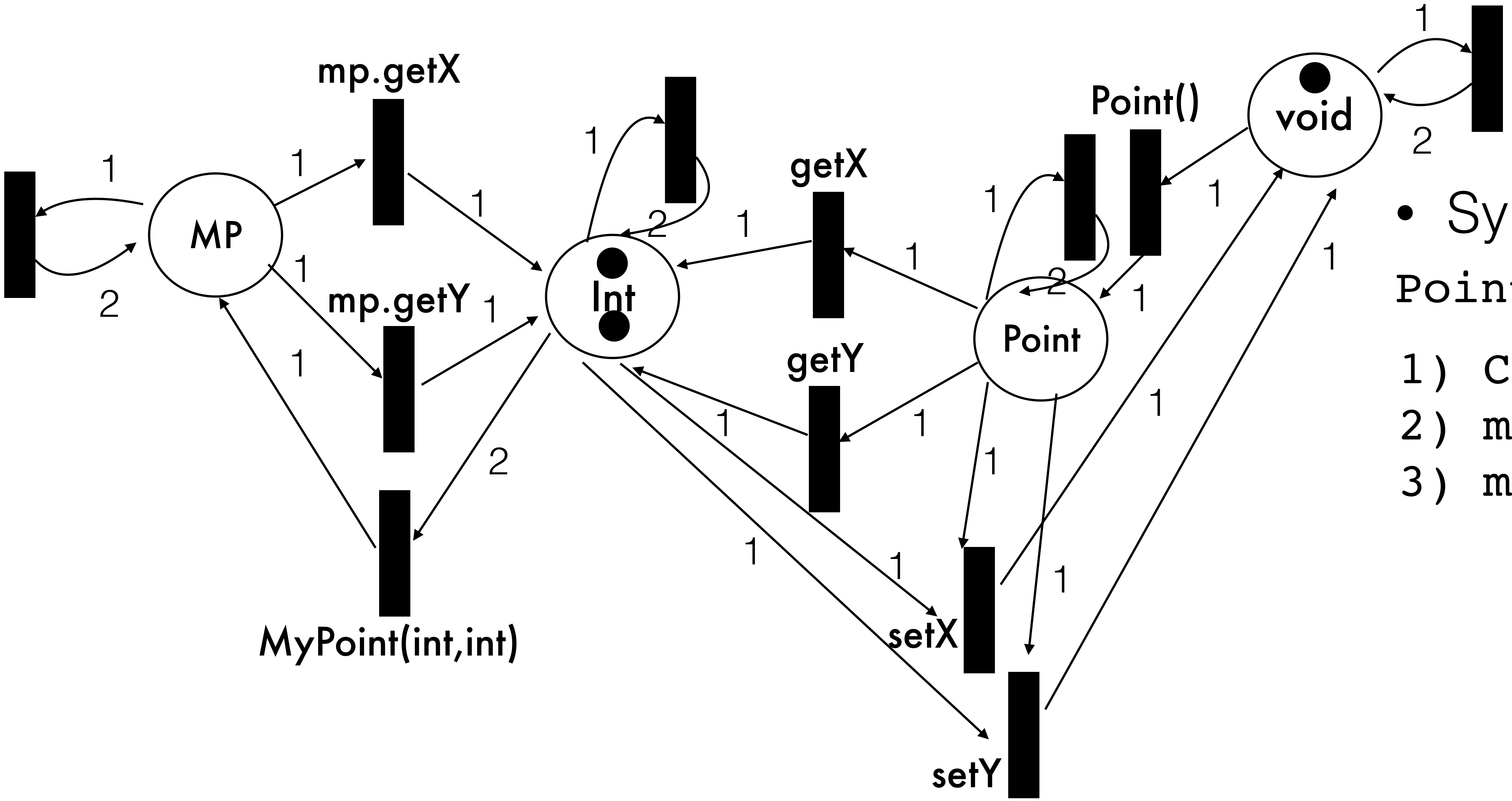


- Synthesize this function:

```
Point convert(Mypoint pt)
```

```
1) Clone-MP
2) mp.getX
```

Exercise 2: Reachable paths

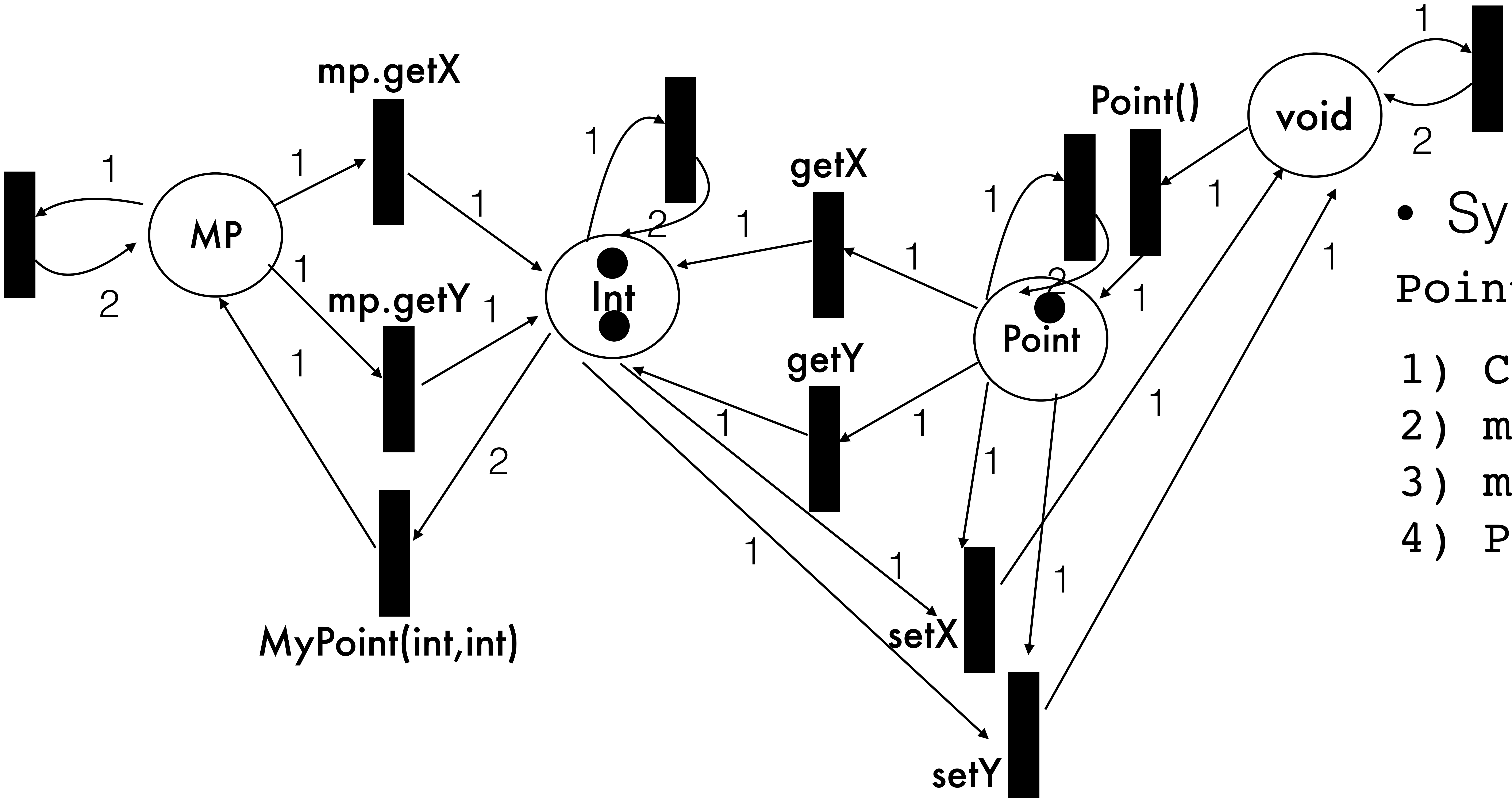


- Synthesize this function:

```
Point convert(Mypoint pt)
```

```
1) Clone-MP
2) mp.getX
3) mp.getY
```

Exercise 2: Reachable paths

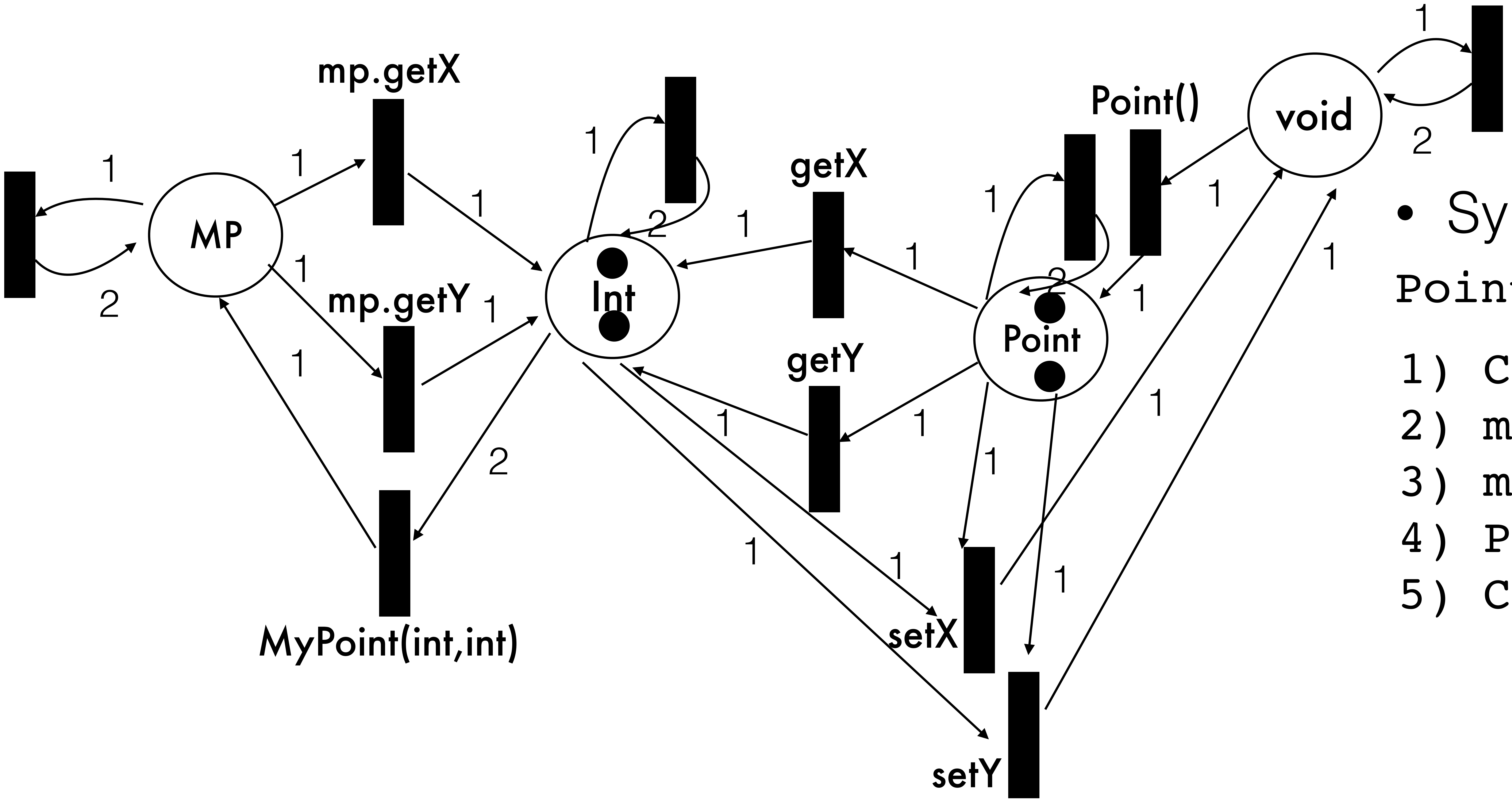


- Synthesize this function:

```
Point convert(Mypoint pt)
```

```
1) Clone-MP
2) mp.getX
3) mp.getY
4) Point()
```

Exercise 2: Reachable paths

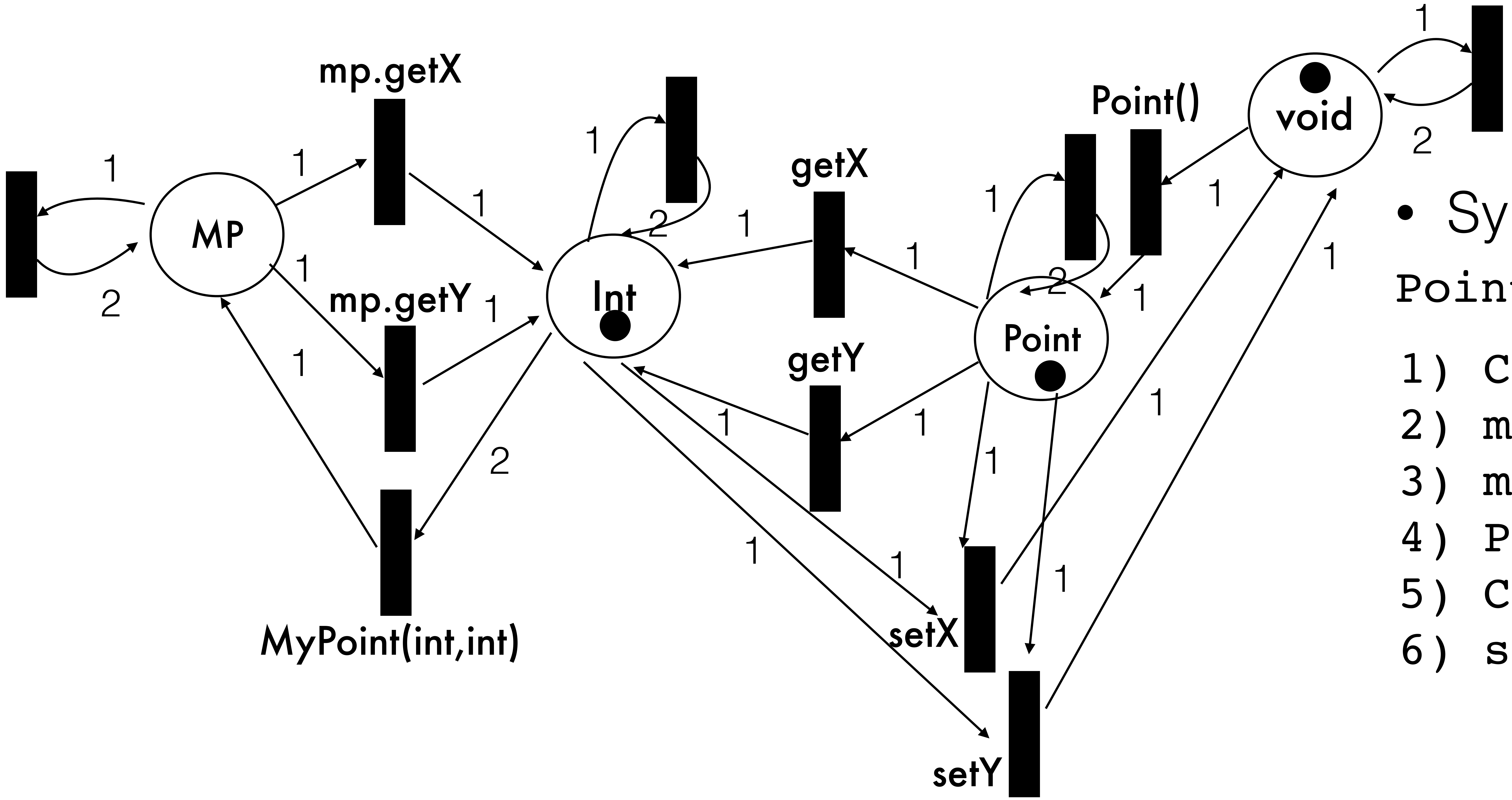


- Synthesize this function:

```
Point convert(Mypoint pt)
```

```
1) Clone-MP
2) mp.getX
3) mp.getY
4) Point()
5) Clone-Point
```

Exercise 2: Reachable paths

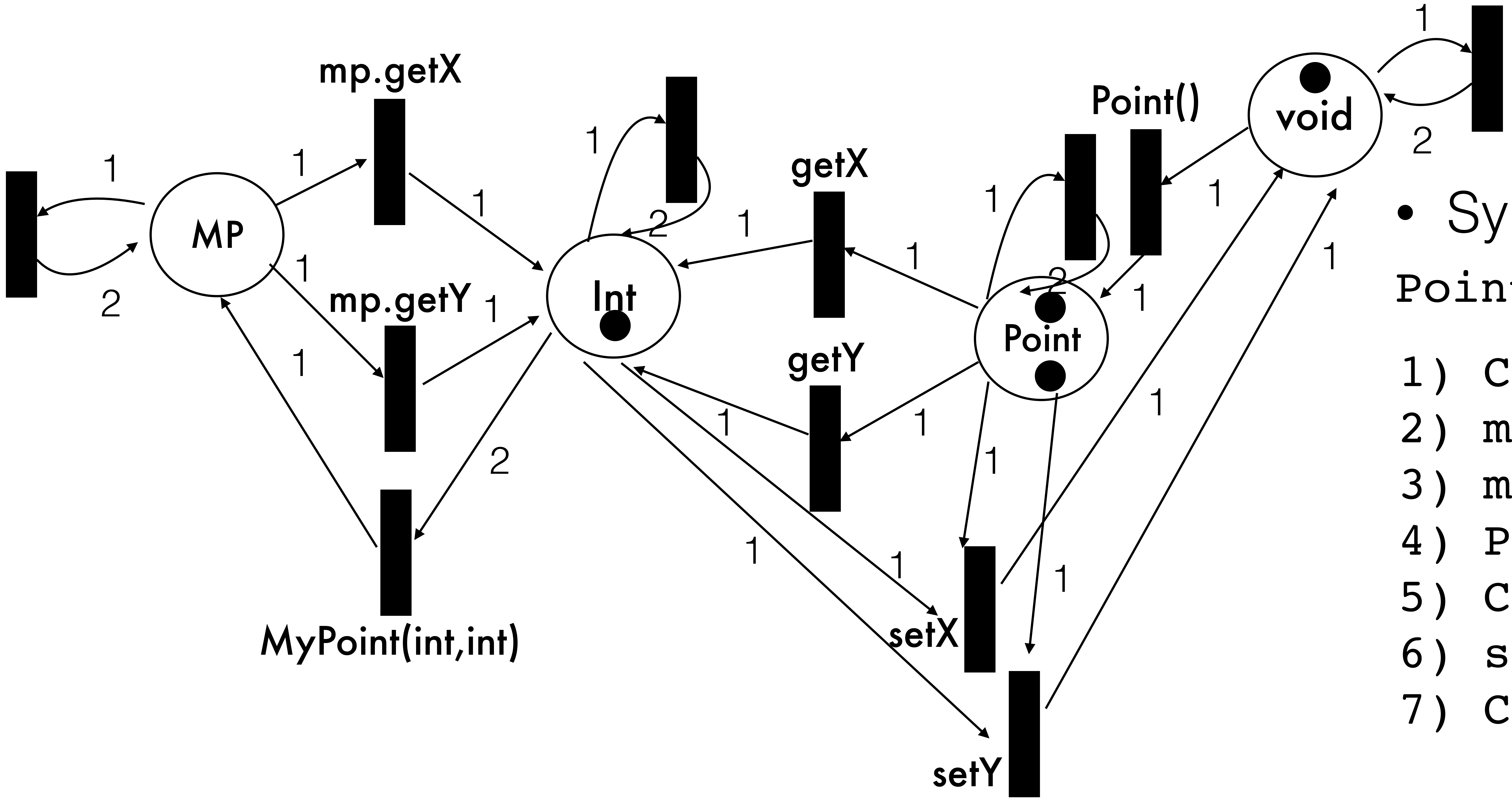


- Synthesize this function:

```
Point convert(Mypoint pt)
```

```
1) Clone-MP
2) mp.getX
3) mp.getY
4) Point()
5) Clone-Point
6) setX
```

Exercise 2: Reachable paths

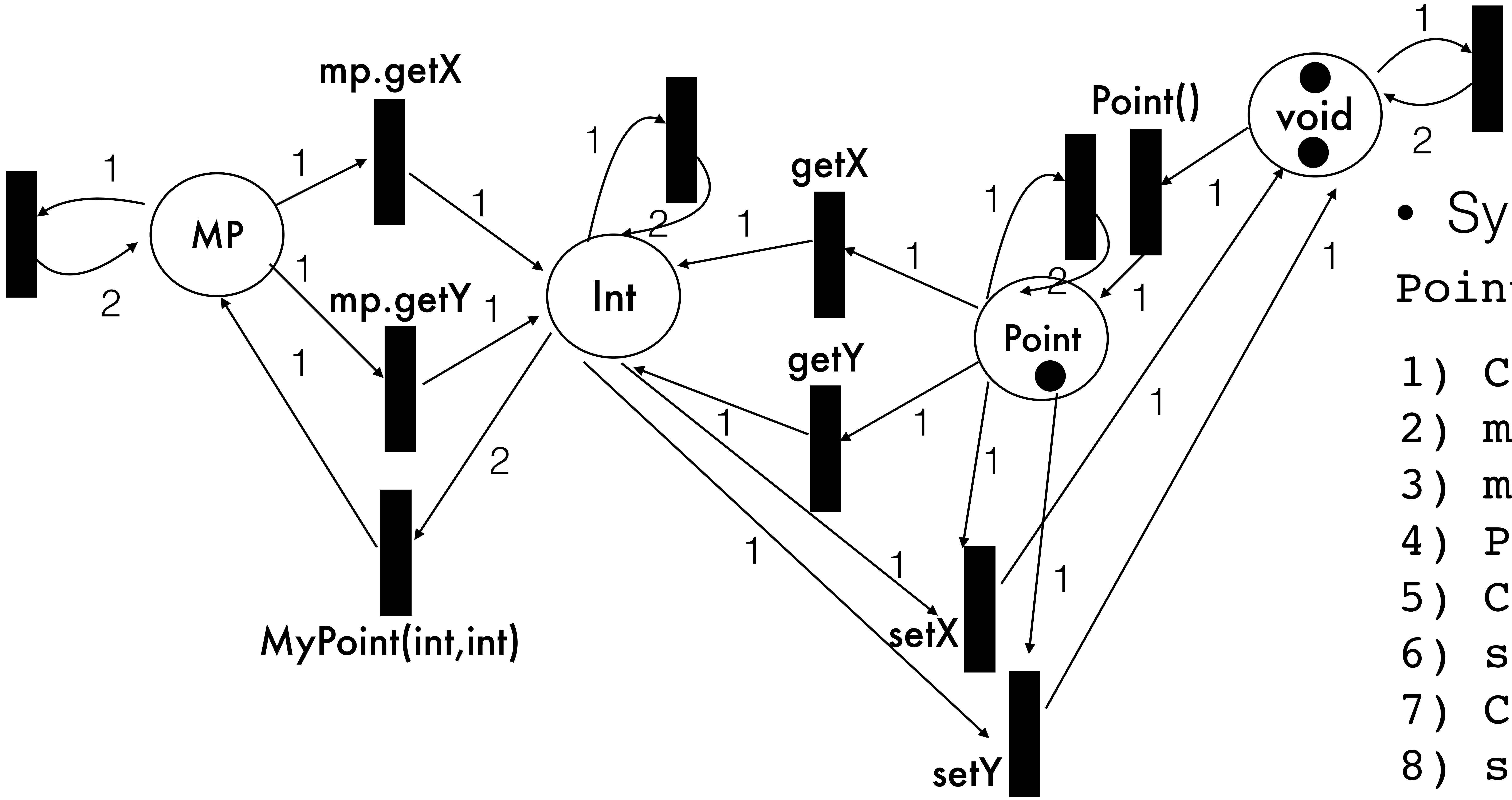


- Synthesize this function:

```
Point convert(Mypoint pt)
```

```
1) Clone-MP
2) mp.getX
3) mp.getY
4) Point()
5) Clone-Point
6) setX
7) Clone-Point
```

Exercise 2: Reachable paths

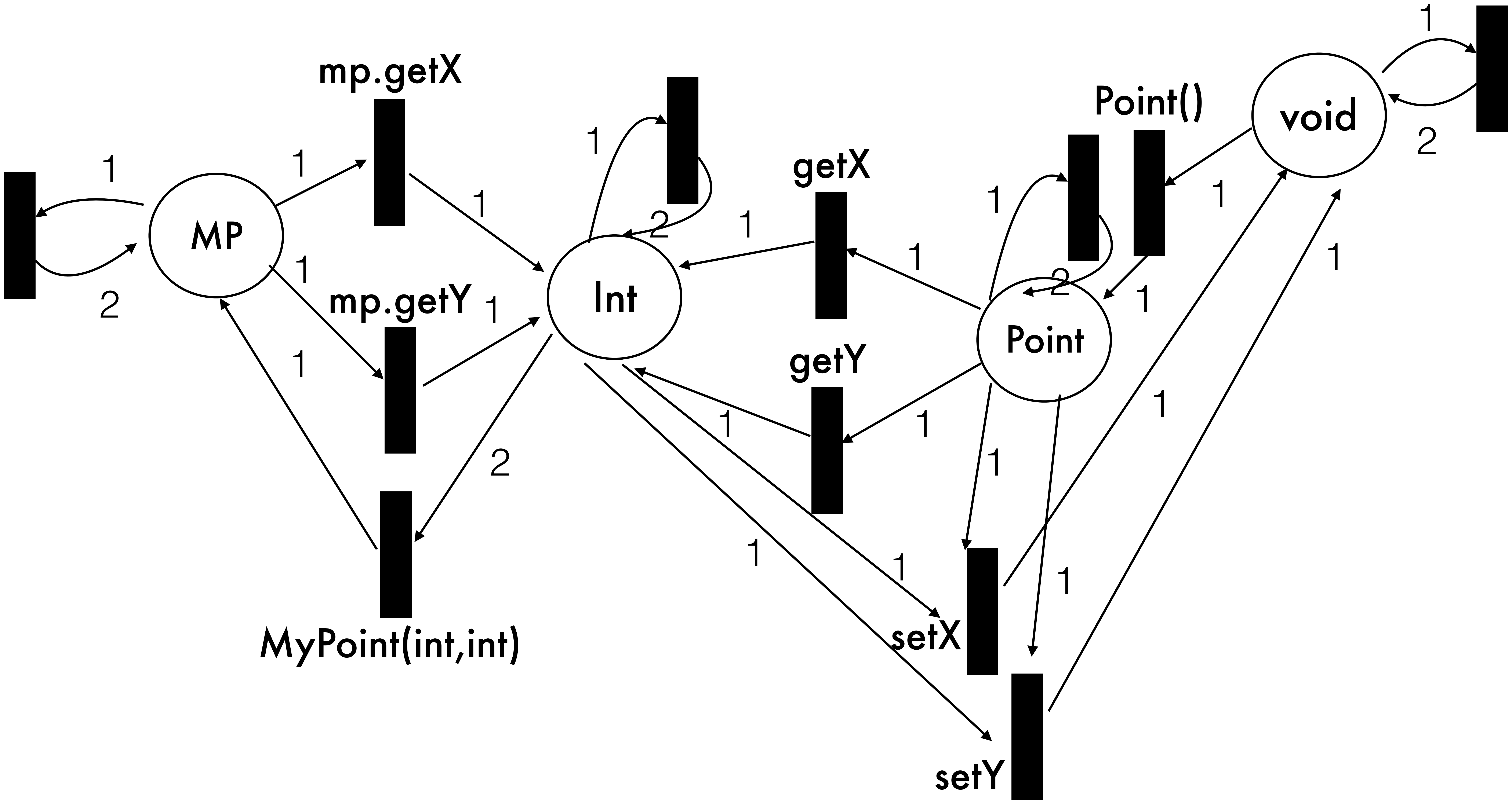


- Synthesize this function:

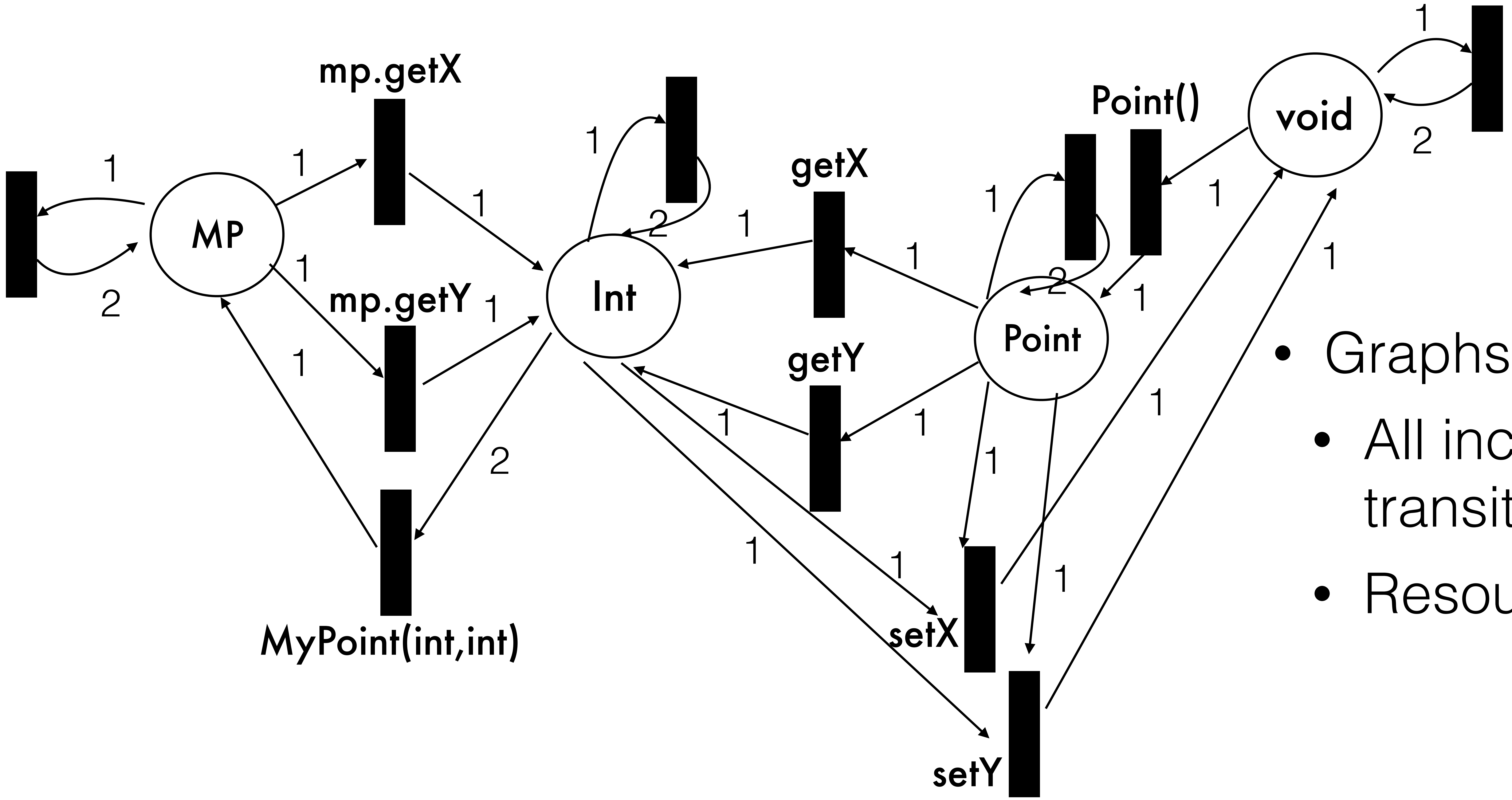
```
Point convert(Mypoint pt)
```

```
1) Clone-MP
2) mp.getX
3) mp.getY
4) Point()
5) Clone-Point
6) setX
7) Clone-Point
8) setY
```

Why a Petri net and not a graph?

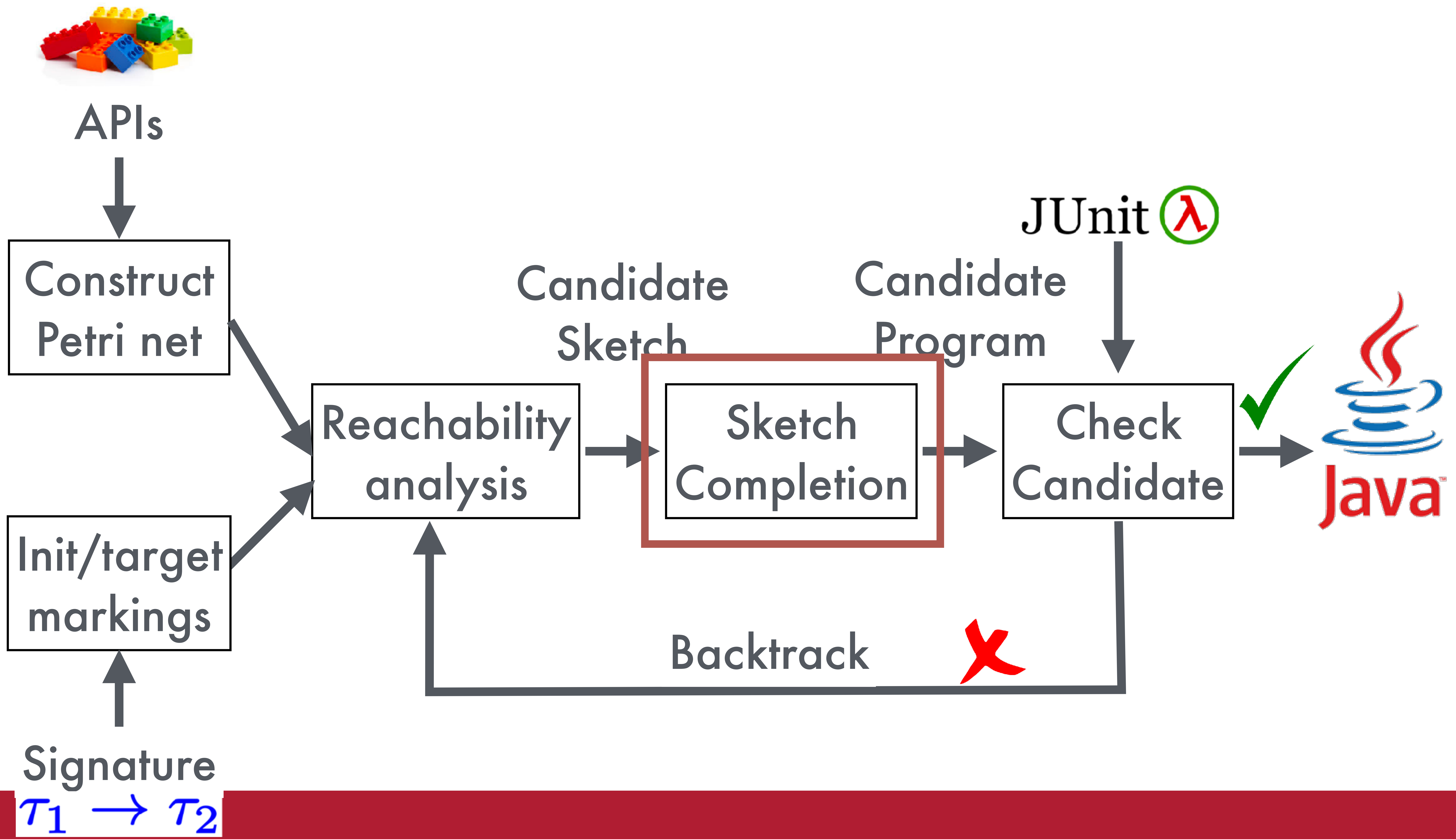


Why a Petri net and not a graph?



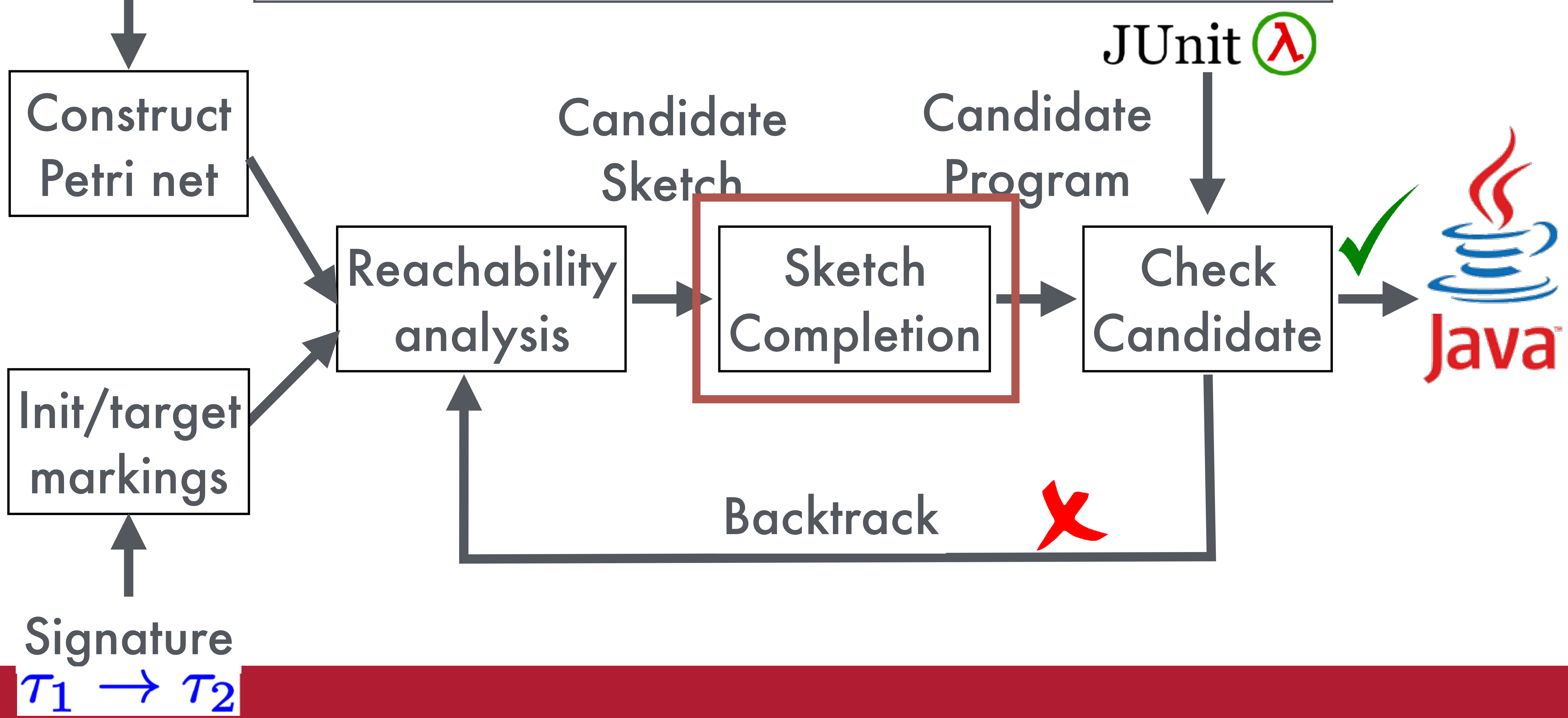
- Graphs do not support:
  - All incoming edges to a transition are part of the path
  - Resource consumption

Accepting run as program sketch



Accepting run as program sketch

Accepting run of Petri net corresponds to program sketch because it does not specify arguments of each method call.



Sketch completion

```
x = #1.getX(); y = #2.getY();
#3.setToRotation(#4, #5, #6);
a = #7.createTransformedArea(#8);
return #9;
```

- Given a path:
  - getX -> getY -> setToRotation -> createTransformedArea
- Find the arguments that should be used in each hole such that the program type checks

Exercise 3: Sketch completion

```
1) Clone-MP
2) mp.getX
3) mp.getY
4) Point()
5) Clone-Point
6) setX
7) Clone-Point
8) setY
```

- Remove the Clone transitions

```
Point convert(Mypoint pt){




}
```

Exercise 3: Sketch completion

```
2) mp.getX
3) mp.getY
4) Point()
6) setX
8) setY
```

```
Point convert(Mypoint pt){




}
```

- What is the code with holes?

Exercise 3: Sketch completion

```
2) mp.getX
3) mp.getY
4) Point()
6) setX
8) setY
```

- What is the code with holes?

```
Point convert(Mypoint pt){

  int x = #1.getX();
  int y = #2.getY();
  Point p = new Point();
  #3.setX(#4);
  #5.setY(#6);
  return #7;

}
```

Exercise 3: Sketch completion

```
2) mp.getX
3) mp.getY
4) Point()
6) setX
8) setY
```

```
Point convert(Mypoint pt){

  int x = pt.getX();
  int y = pt.getY();
  Point p = new Point();
  p.setX(#4);
  p.setY(#6);
  return p;

}
```

- What is the code with holes?

Exercise 3: Sketch completion

```
2) mp.getX
3) mp.getY
4) Point()
6) setX
8) setY
```

- What is the code with holes?

```
Point convert(Mypoint pt){

  int x = pt.getX();
  int y = pt.getY();
  Point p = new Point();
  p.setX(y);
  p.setY(x);
  return p;

}
```

Exercise 3: Sketch completion

```
2) mp.getX
3) mp.getY
4) Point()
6) setX
8) setY
```

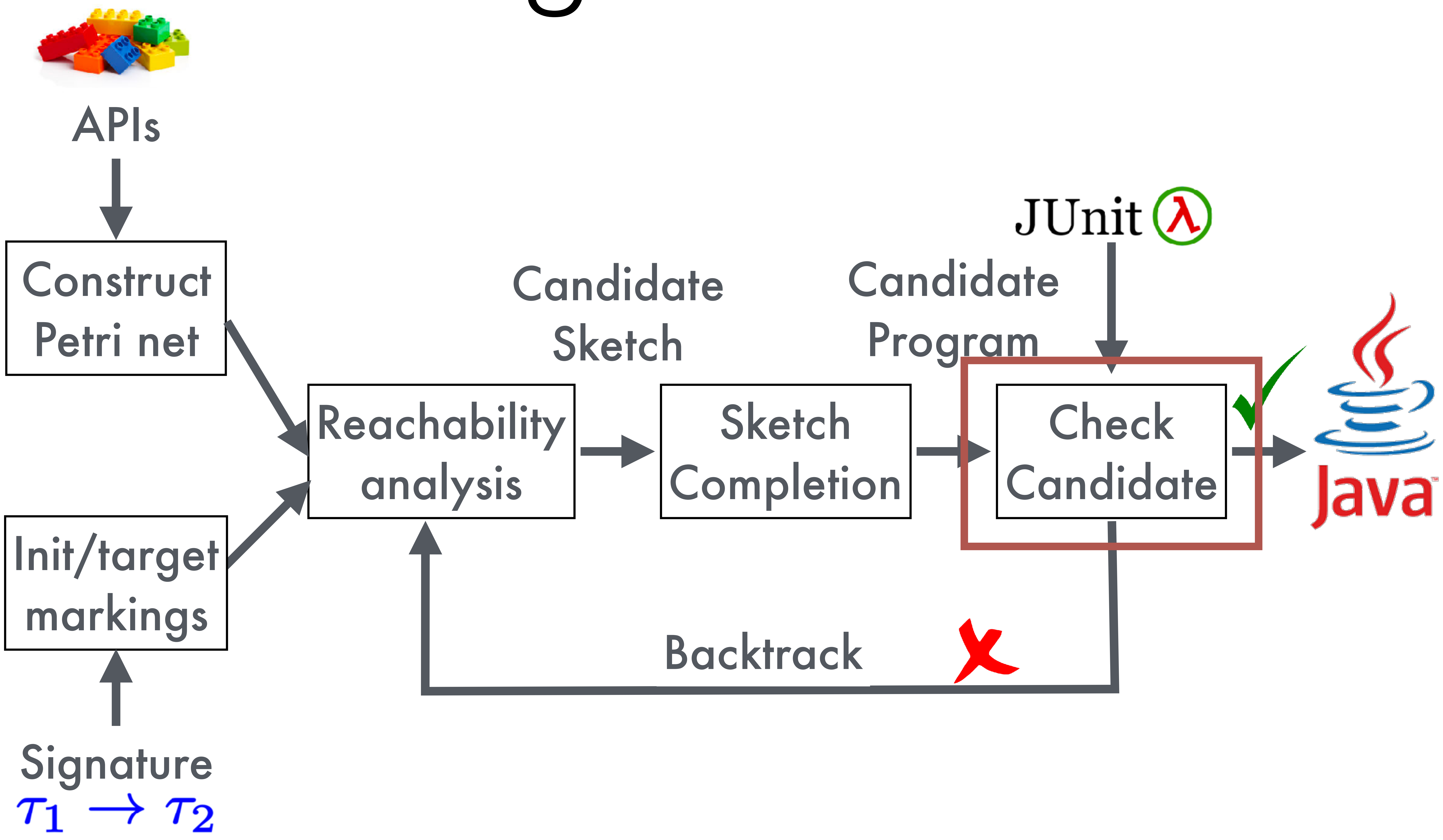
```
Point convert(Mypoint pt){

  int x = pt.getX();
  int y = pt.getY();
  Point p = new Point();
  p.setX(x);
  p.setY(y);
  return p;

}
```

- What is the code with holes?

Checking the candidate



Test cases

```
Point convert(Mypoint pt){

  int x = pt.getX();
  int y = pt.getY();
  Point p = new Point();
  p.setX(x);
  p.setY(y);
  return p;

}
```

- Write a test case to check the conversion

Test cases

```
Point convert(Mypoint pt){

  int x = pt.getX();
  int y = pt.getY();
  Point p = new Point();
  p.setX(x);
  p.setY(y);
  return p;

}
```

- Write a test case to check the conversion

```
bool test(){

  MyPoint mp = new MyPoint(1,2);
  Point p = convert(mp);
  return (p.getX() == 1 &&
          p.getY() == 2);

}
```

How does relate with SAT?

- Petri net reachability can encoded into SAT

- SyPet encodes this problem to SAT by viewing it as a **bounded** planning problem in STRIPS:
  - Variables
  - Actions: preconditions -> postconditions
  - Initial State
  - Goal State

How does relate with SAT?

- **Variables:**
  - Consider a fixed number of time steps *t*
  - For each place (type), let *m* be the maximum number of tokens that can occur in that place:
    - Create *t * m* variables: x(place,t’,m)
    - Example: x(int,0,1) denotes that place *int* contains 1 token at time step *0*
  - For each transition (API):
    - Create *t* variables: x(api, t’)
    - Example: x(getX, 1) means that *getX* is fired at time step *1*

How does relate with SAT?

- **Constraints**
  - Exactly one transition is fired at each time step t'
  - ***(preconditions)*** A transition *f* can only be fired if there are enough tokens in the places from its incoming places
  - ***(postconditions)*** If a transition *f* is fired at time step *t'* then:
    - All outgoing places from *f* will have their markings increased at time step *t'+1*
    - All incoming places to *f* will have their marking decreased at time step *t'+1*

How does relate with SAT?

- **Constraints**
  - ***(frame axioms)*** If all outgoing transitions *f* from a given place *p* are not fired at time step *t'* then the marking in p does not change between time step *t* 'and *t'+1*
  - ***(initial state)*** The initial state is defined by fixing the initial markings in the respective places at time step *0*
  - ***(goal state)*** The goal state is defined by fixing the final marking in the respective places at time step *t*

Using SyPet

Demo

SyPet's strengths

- Works for real code!
- Generic: can tackle any Java library
  - e.g. geometry, math, joda, unirest, xml, etc.
- Works well when there are many different types
- Scales to a large number of APIs

SyPet’s weaknesses

- Does not support conditionals
- Does not support loops
- For some applications it is hard to write test cases
- Does not scale when everything is the same type

For more information

SyPet

Program synthesis tool for Java libraries that automatically constructs programs by composing APIs.

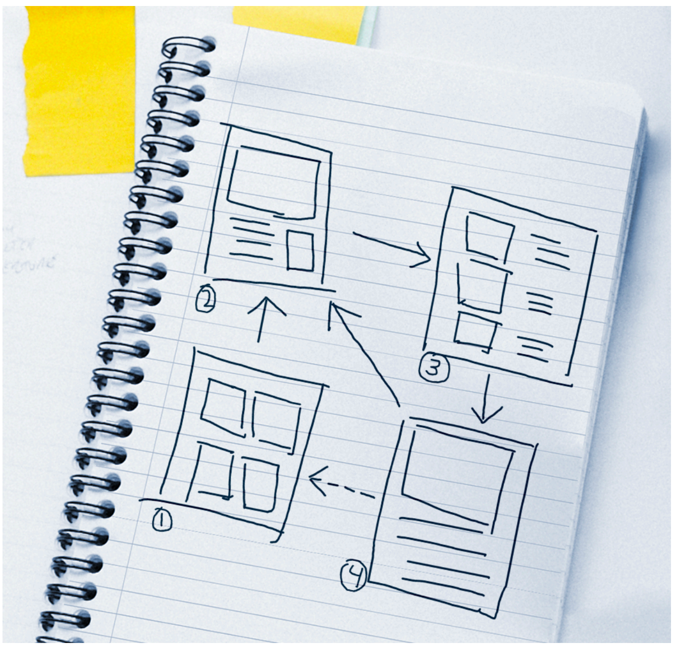
GITHUB DOWNLOAD

https://utopia-group.github.io/sypet/

Outline

Code Reuse	**Complex Java APIs**	**Program Sketching**	**CEGIS**	**SyGuS**
FSE'16	POPL'17	PLDI'05	PLDI'08	FMCAD'13






Program Sketching

```
void doubleSketch(int x){
  int t = x * ??;
  assert t == x + x;
}
```

- Program Sketching:
  - The user provides a partial program with holes
  - The synthesizer finds an assignment to the holes such that the specification is satisfied

Program Sketching

```
void doubleSketch(int x){
  int t = x * ??;
  assert t == x + x;
}
```

```
void doubleSketch(int x){
  int t = x * 2;
  assert t == x + x;
}
```

- Program Sketching:
  - The user provides a partial program with holes
  - The synthesizer finds an assignment to the holes such that the specification is satisfied

Program Sketching

Demo

- For more details:
  - https://people.csail.mit.edu/asolar/
  - Manual: https://people.csail.mit.edu/asolar/manual.pdf

Outline

Code Reuse

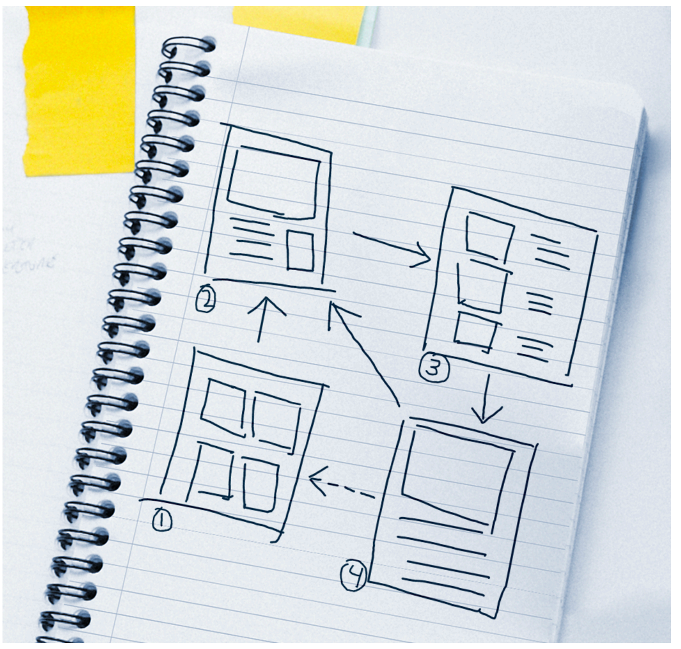
FSE'16

Complex Java APIs

POPL'17

Program Sketching

PLDI'05



CEGIS

PLDI'08

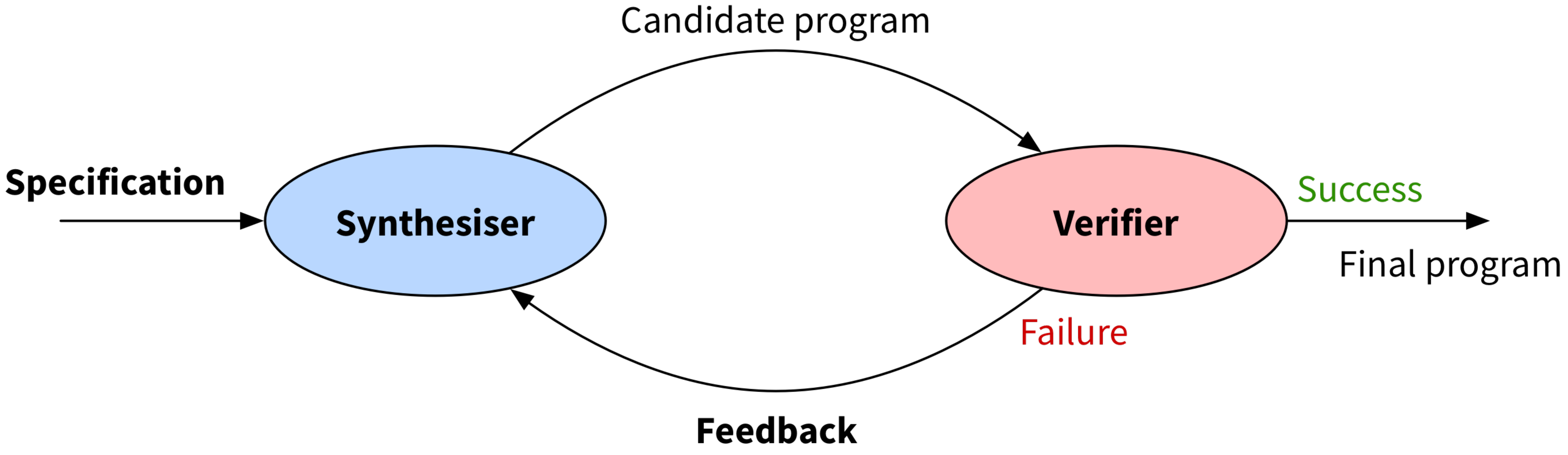
SyGuS

FMCAD'13

SyGuS
Syntax-Guided Synthesis

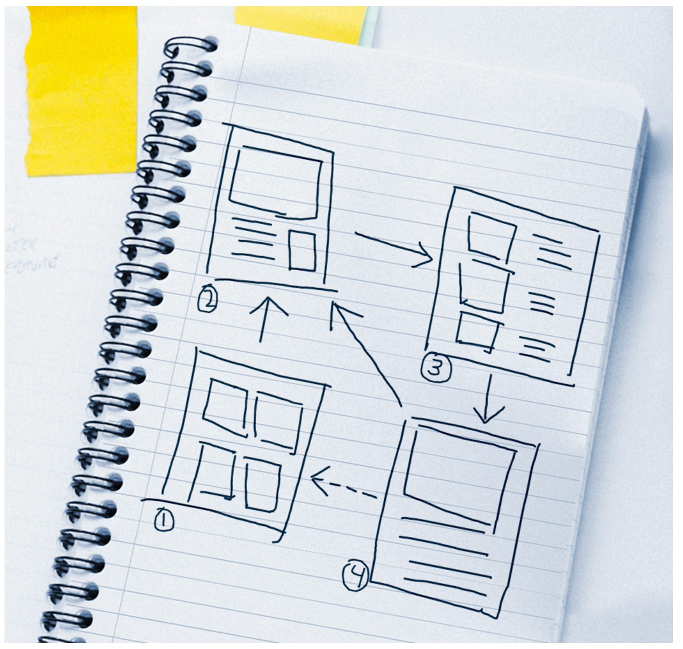
Counterexample-Guided Inductive Synthesis

CEGIS:



Outline

Code Reuse	Complex Java APIs	Program Sketching	CEGIS	SyGuS
FSE’16	POPL’17	PLDI’05	PLDI’08	FMCAD’13



SyGuS
Syntax-Guided Synthesis

Syntax-Guided Synthesis

- Extends the SMT-Lib language to synthesis
- Advantages:
  - Same input format that can be reused by solvers
  - Easy to specify for users
- Disadvantages:
  - Limited to SMT theories

Syntax-Guided Synthesis

```
;; The background theory is linear integer arithmetic
(set-logic LIA)

;; Name and signature of the function to be synthesized
(synth-fun max2 ((x Int) (y Int)) Int

    ;; Declare the non-terminals that would be used in the grammar
    ((I Int) (B Bool))

    ;; Define the grammar for allowed implementations of max2
    ((I Int (x y 0 1
             (+ I I) (- I I)
             (ite B I I)))
     (B Bool ((and B B) (or B B) (not B)
              (= I I) (<= I I) (>= I I))))
)

(declare-var x Int)
(declare-var y Int)

;; Define the semantic constraints on the function
(constraint (>= (max2 x y) x))
(constraint (>= (max2 x y) y))
(constraint (or (= x (max2 x y)) (= y (max2 x y))))

(check-synth)
```

Syntax-Guided Synthesis

- For more details and solvers:
  - https://sygus.org/

Papers

FSE'16	Hunter: Next-Generation Code Reuse for Java
POPL'17	Component-Based Synthesis for Complex APIs
PLDI'05	Programming by sketching for bit-streaming programs
PLDI'08	Sketching concurrent data structures
FMCAD'13	Syntax-guided synthesis