




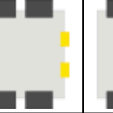
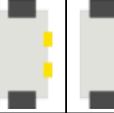





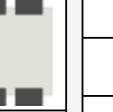






Test structurel

Exemple du robot

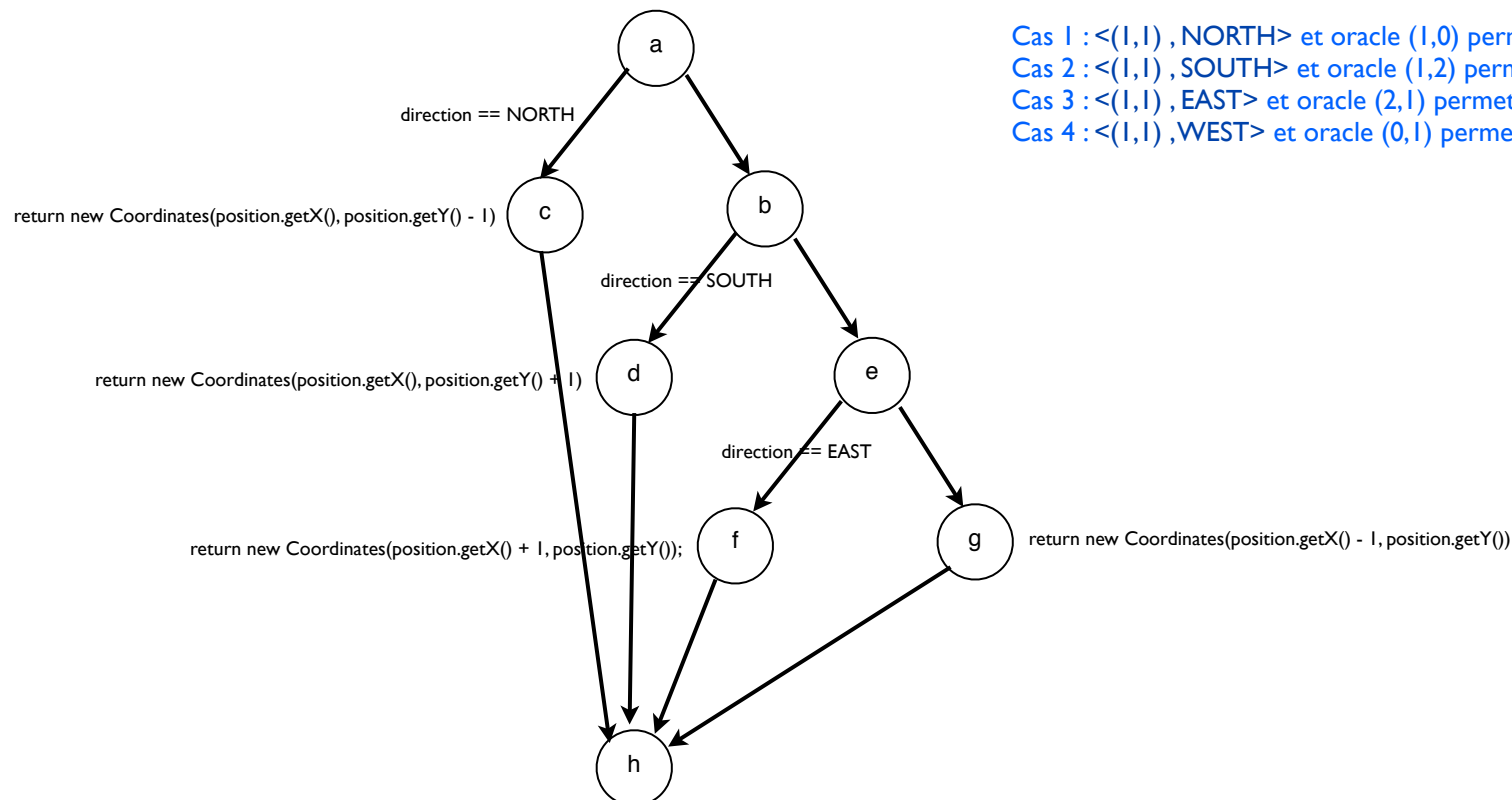
Fabrice AMBERT – fabrice.ambert@femto-st.fr
Fabrice BOUQUET – fabrice.bouquet@femto-st.fr
Fabien PEUREUX – fabien.peureux@femto-st.fr
Ivan ENDERLIN, Jean-Marie GAUTHIER
Cédric JOFFROY, Alexandre VERNOTTE

Y \ X	1	2	3	4	5	6	7	8	9	10	Ordre	Arrivée
1											↑	(10,9,nord)
											↑	(10,8,nord)
2											←	(10,8,ouest)
											↑	(9,8,ouest)
3											↑	(8,8,ouest)
											↑	(7,8,ouest)
4											→	(7,8,nord)
											↑	(7,7,nord)
5											↑	(7,6,nord)
											→	(7,6,est)
6											↓	(6,6,est)
											↓	(5,6,est)
7											↓	(4,6,est)
											↓	(3,6,est)
8											←	(3,6,nord)
											↑	(3,5,nord)
9											↑	(3,4,nord)
											↑	(3,3,nord)
10												

nextForwardPosition

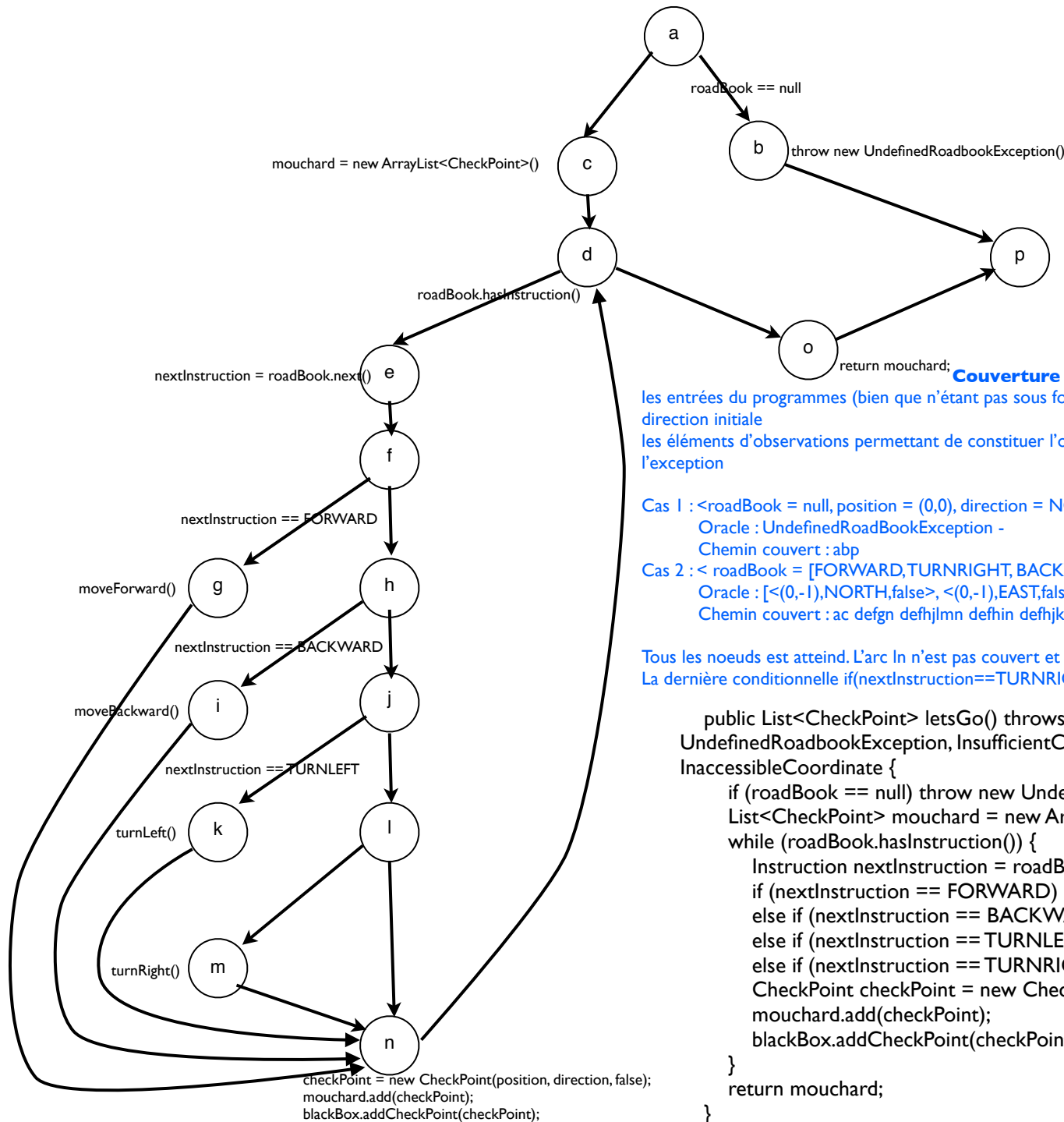
```
static Coordinates nextForwardPosition(Coordinates position, Direction direction) {
    if (direction == NORTH)
        return new Coordinates(position.getX(), position.getY() - 1);
    if (direction == SOUTH)
        return new Coordinates(position.getX(), position.getY() + 1);
    if (direction == EAST)
        return new Coordinates(position.getX() + 1, position.getY());
    return new Coordinates(position.getX() - 1, position.getY());
}
```

Sur ce graphe, quelque soit le critère de couverture choisi parmi «tous nœuds», «tous arcs» ou «tous chemins indépendants», 4 cas de tests suffisent à obtenir de critère. Le paramètre direction permet de fixer le chemin couvert et le paramètre position d'établir l'oracle.



Cas 1 : $\langle (1,1), \text{NORTH} \rangle$ et oracle $(1,0)$ permet de couvrir ach
 Cas 2 : $\langle (1,1), \text{SOUTH} \rangle$ et oracle $(1,2)$ permet de couvrir abdh
 Cas 3 : $\langle (1,1), \text{EAST} \rangle$ et oracle $(2,1)$ permet de couvrir abefh
 Cas 4 : $\langle (1,1), \text{WEST} \rangle$ et oracle $(0,1)$ permet de couvrir abegh

letsGo



Couverture de tous les noeuds

les entrées du programmes (bien que n'étant pas sous forme de paramètres) sont : le roadBook, la position initiale et la direction initiale

les éléments d'observations permettant de constituer l'oracle sont : la liste de checkPoint retourné par la méthode ou l'exception

Cas 1 : <roadBook = null, position = (0,0), direction = NORTH> -

Oracle : UndefinedRoadbookException -

Chemin couvert : abp

Cas 2 : < roadBook = [FORWARD,TURNRIGHT, BACKWARD,TURNLEFT], position = (0,0), direction = NORTH> -

Oracle : [<(0,-1),NORTH,false>, <(0,-1),EAST,false>, <(-1,-1),EAST,false>, <(-1,-1),NORTH,false>] -

Chemin couvert : ac defgn defhijlmn defhijn dop

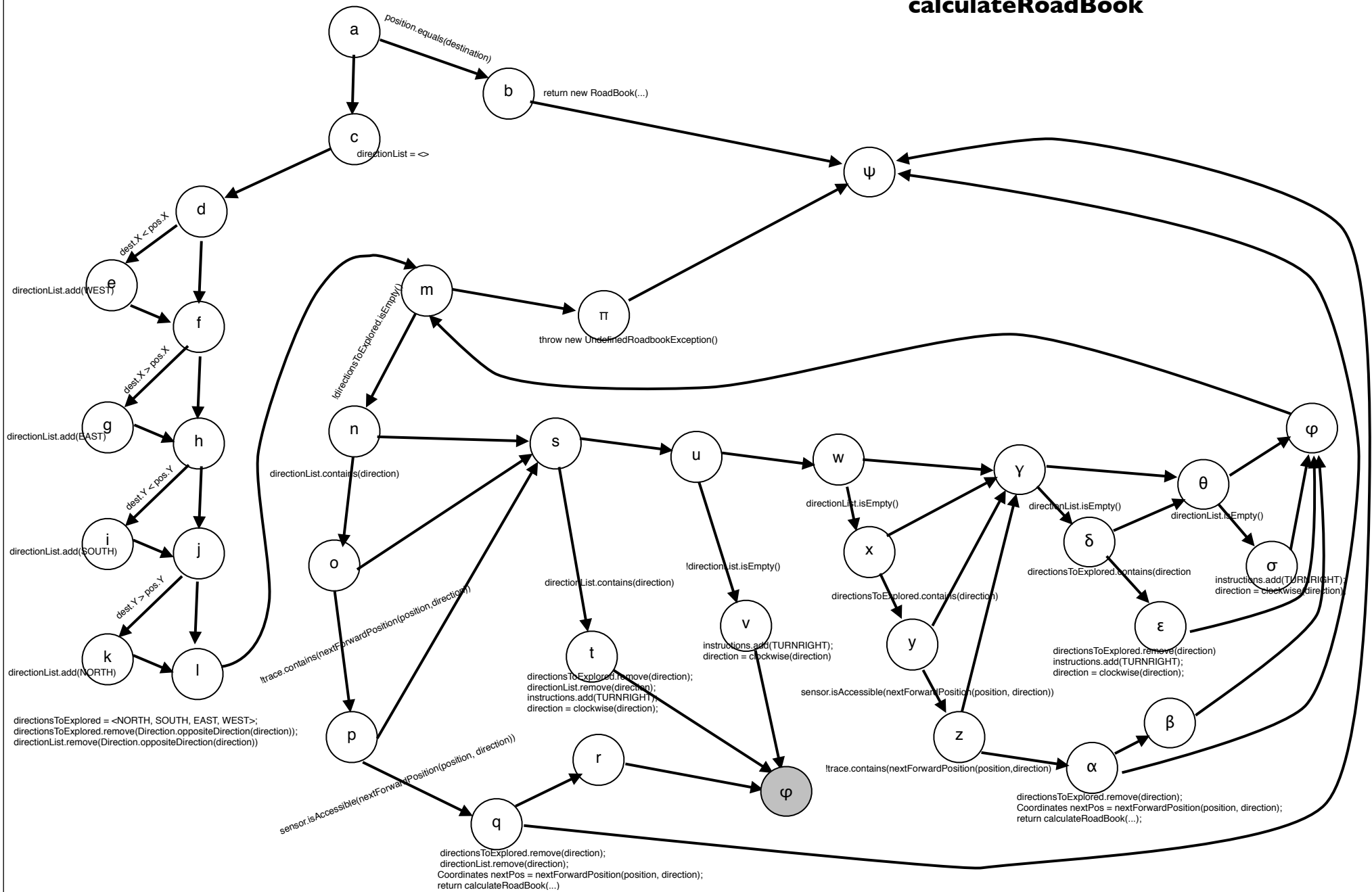
Tous les noeuds est atteint. L'arc ln n'est pas couvert et il n'existe pas de valeur d'instruction permettant de le couvrir. La dernière conditionnelle if(nextInstruction==TURNRIGHT) est inutile.

```

public List<Checkpoint> letsGo() throws UnlandedRobotException,
UndefinedRoadbookException, InsufficientChargeException, LandSensorDefaillance,
InaccessibleCoordinate {
    if (roadBook == null) throw new UndefinedRoadbookException();
    List<Checkpoint> mouchard = new ArrayList<Checkpoint>();
    while (roadBook.hasNextInstruction()) {
        Instruction nextInstruction = roadBook.next();
        if (nextInstruction == FORWARD) moveForward();
        else if (nextInstruction == BACKWARD) moveBackward();
        else if (nextInstruction == TURNLEFT) turnLeft();
        else if (nextInstruction == TURNRIGHT) turnRight();
        CheckPoint checkPoint = new CheckPoint(position, direction, false);
        mouchard.add(checkPoint);
        blackBox.addCheckpoint(checkPoint);
    }
    return mouchard;
}
  
```



calculateRoadBook



Mc Cabe : 21, à l'évidence cette méthode nécessite un effort de reconception pour réduire taille du graphe de flot de contrôle

CalculateRoadBook après quelques réorganisations

```
static RoadBook calculateRoadBook(LandSensor sensor, Direction direction, Coordinates position, Coordinates destination, List<Instruction> instructions, List<Coordinates> trace) throws LandSensorDefaillance, UndefinedRoadbookException {
    if (position.equals(destination)) return new RoadBook(InstructionListTool.compacte(instructions));
    List<Direction> privilegeDirections = computePrivilegeDirection(position, destination);
    List<Direction> directionsToExplored = new ArrayList<Direction>(Arrays.asList(NORTH, SOUTH, EAST, WEST));
    directionsToExplored.remove(Direction.oppositeDirection(direction));
    privilegeDirections.remove(Direction.oppositeDirection(direction));
    while (!directionsToExplored.isEmpty()) {
        if (privilegeDirections.contains(direction) || privilegeDirections.isEmpty()) {
            privilegeDirections.remove(direction);
            if (directionsToExplored.contains(direction)) {
                directionsToExplored.remove(direction);
                Coordinates nextPos = nextForwardPosition(position, direction);
                if (sensor.isAccessible(nextPos) && !trace.contains(nextPos)) {
                    try {
                        return calculateRoadBook(sensor, direction, nextPos, destination, concatene(instructions, FORWARD), concatene(trace, nextPos));
                    } catch (UndefinedRoadbookException e) {
                    }
                }
            }
        }
    }

    instructions.add(TURNRIGHT);
    direction = clockwise(direction);
}

throw new UndefinedRoadbookException();
}

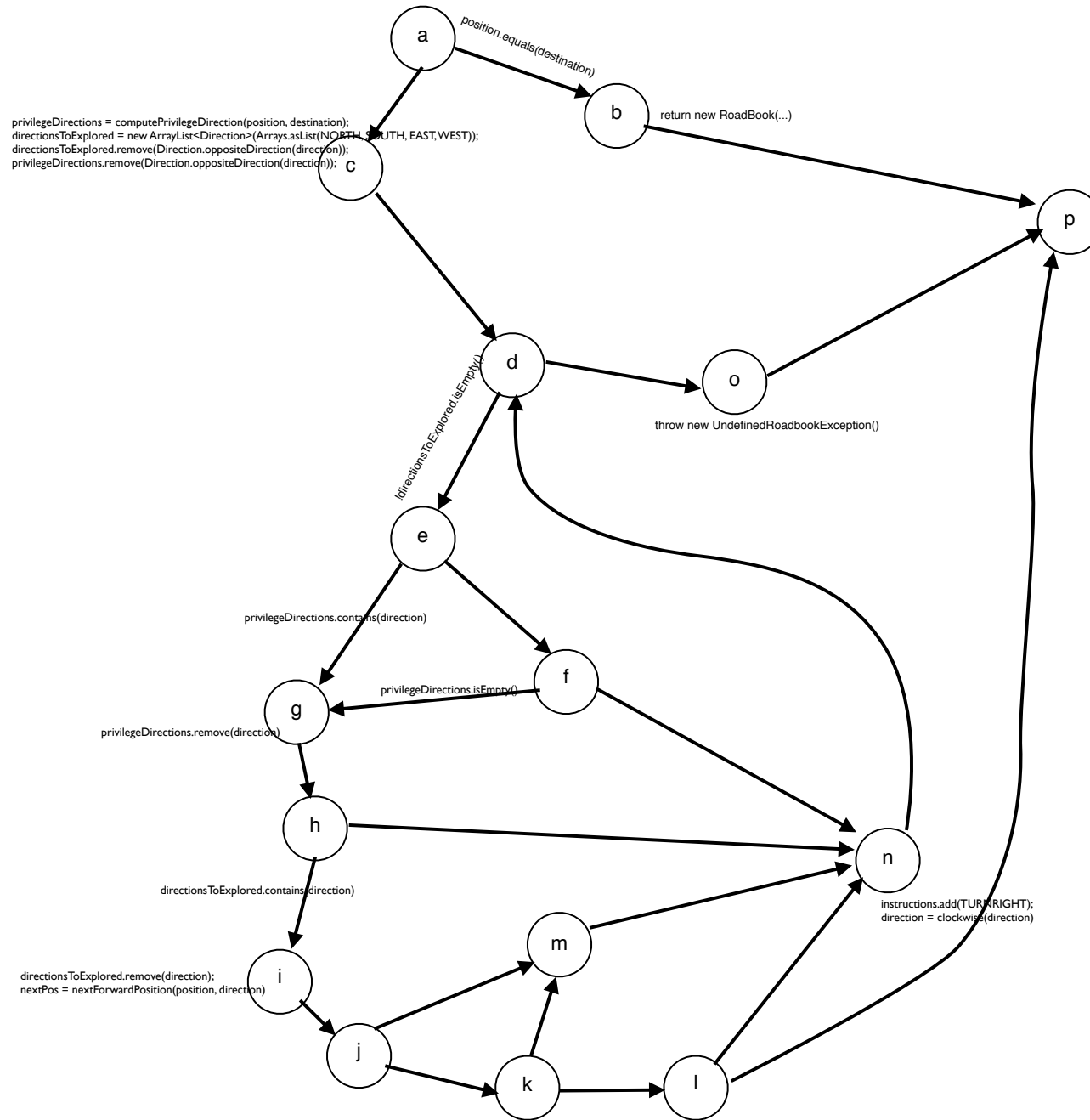
static List<Direction> computePrivilegeDirection(Coordinates position, Coordinates destination) {
    List<Direction> privilegeDirections = new ArrayList<Direction>();
    if (destination.getX() < position.getX()) privilegeDirections.add(WEST);
    if (destination.getX() > position.getX()) privilegeDirections.add(Direction.EAST);
    if (destination.getY() > position.getY()) privilegeDirections.add(Direction.SOUTH);
    if (destination.getY() < position.getY()) privilegeDirections.add(Direction.NORTH);
    return privilegeDirections;
}
```

On réorganise les conditions pour éviter les tests redondants dans les branches des if else if...

On extrait une méthode pour déterminer les directions privilégiées

Ceci ne résout cependant pas les problèmes d'explosion combinatoire et de consommation mémoire

calculateRoadBook



Mc Cabe : 9 ce qui rend le test de cette version plus abordable que la précédente