Model based test generation for web apps

Paolo Tonella

Fondazione Bruno Kessler, Trento, Italy

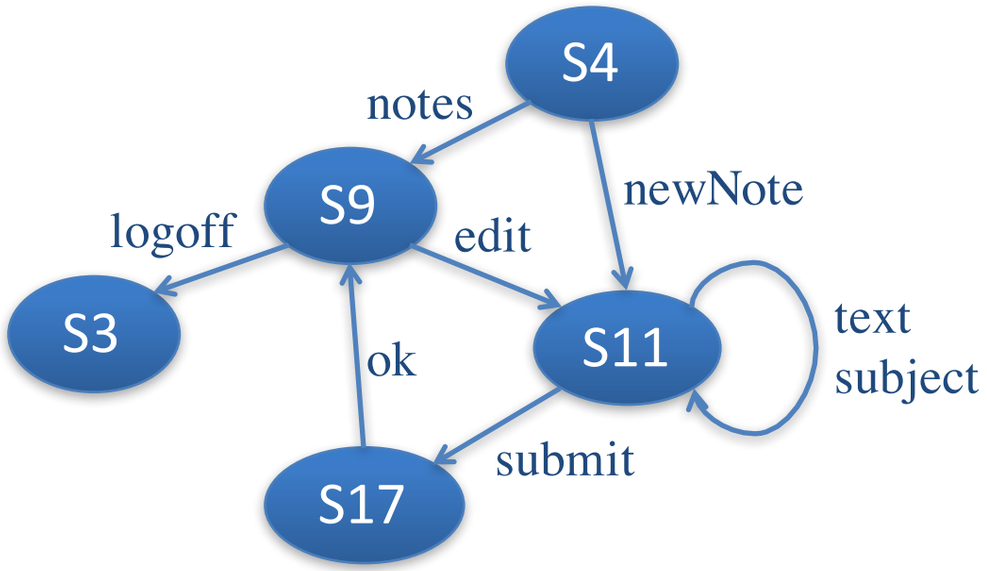
tonella@fbk.eu

Outline

- Model based testing of web apps
- Inferred models and N-grams
  - Paolo Tonella, Roberto Tiella, Cu Duy Nguyen, *Interpolated n-grams for model based testing*. ICSE, pp. 562-572, 2014
- *Page Object* and search based test generation
  - Work in progress with Matteo Biagiola and Filippo Ricca

Model based testing

- Behaviour is abstracted as states/transitions
- Test cases are paths satisfying (state/transition) coverage

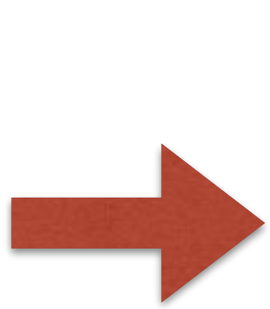


TC1: <newNote, submit, ok, logoff>
TC2: <notes, edit, text, subject, submit>

Model inference

Where do we get the model from?

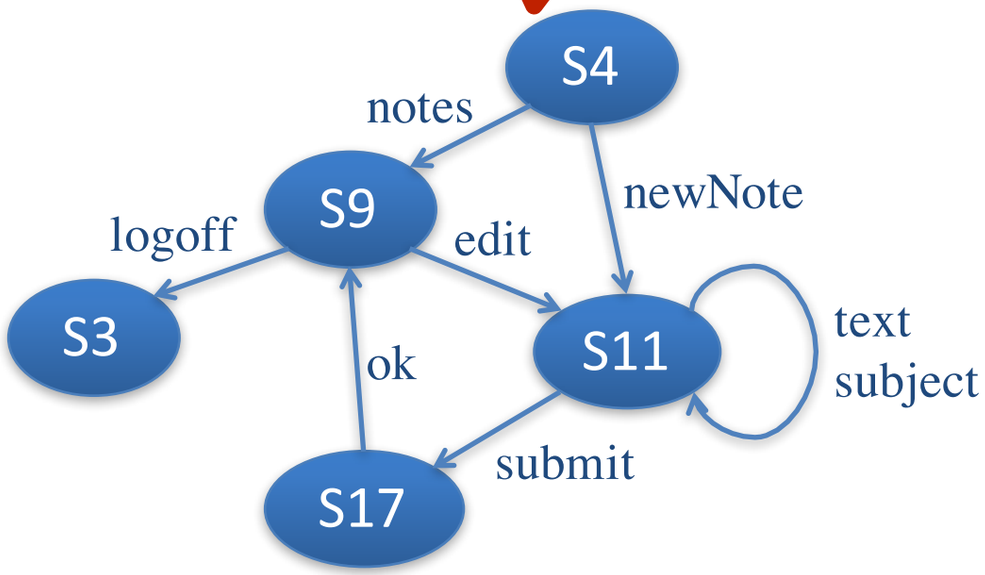
Web crawler



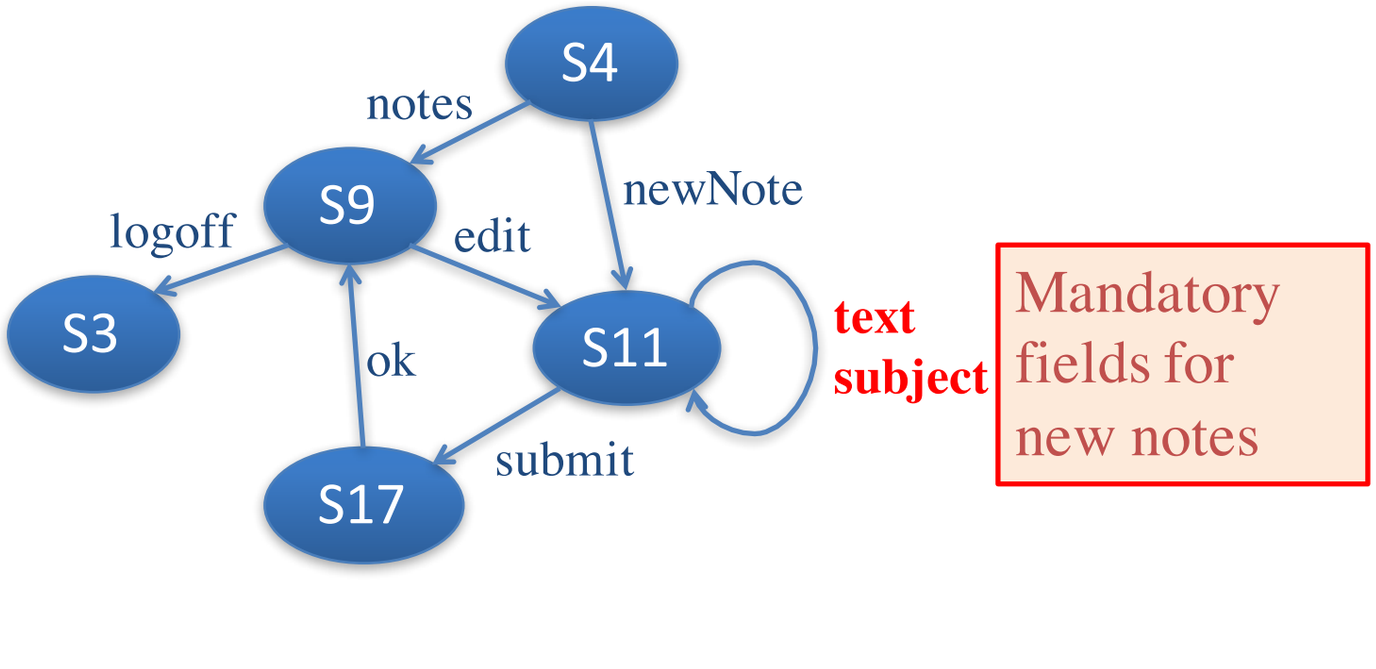
Web pages

State abstraction

State abstraction



Path infeasibility

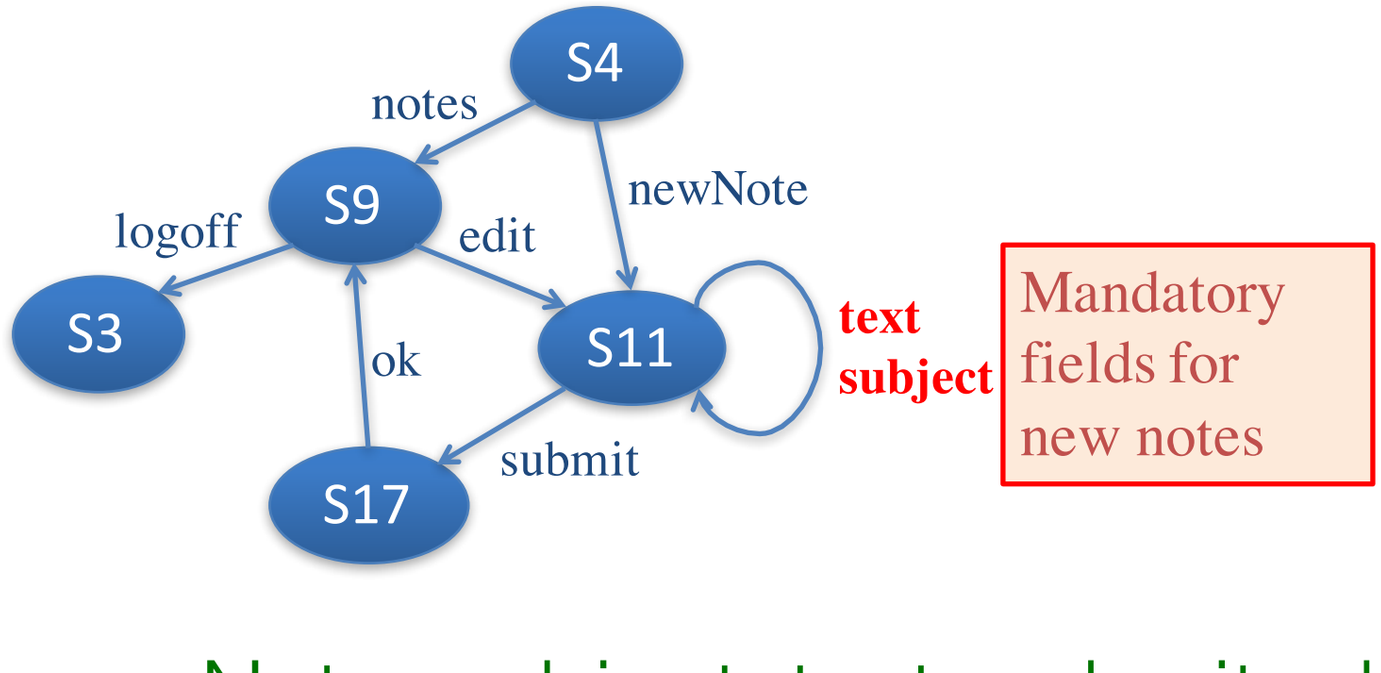


Mandatory fields for new notes

❌ TC1: <newNote, submit, ok, logoff>

✔ TC2: <notes, edit, text, subject, submit>

N-grams for path feasibility



Mandatory fields for new notes

- ✓ TC1: <newNote, subject, text, submit, ok, logoff>
- ✓ TC2: <notes, edit, text, subject, submit>

$P(e_N \mid e_1, \ldots, e_{N-1})$

P(submit | newNote) = 0
P(subject | newNote) = 0.8
P(text | newNote) = 0.2

P(text | subject) = 0.9
P(subject | subject) = 0.07
P(submit | subject) = 0.03

N-gram statistics computed from monitored data (execution logs)

Interpolated N-grams

Problem: *with longer context N, feasibility is more likely, but for a given prefix, no N-tuple might be available if N is too long*

$$P^*(e \mid e_1,\ldots,e_{N-1}) = \alpha \sum_{k=1}^{N-1} 2^k P(e \mid e_1,\ldots,e_k)$$

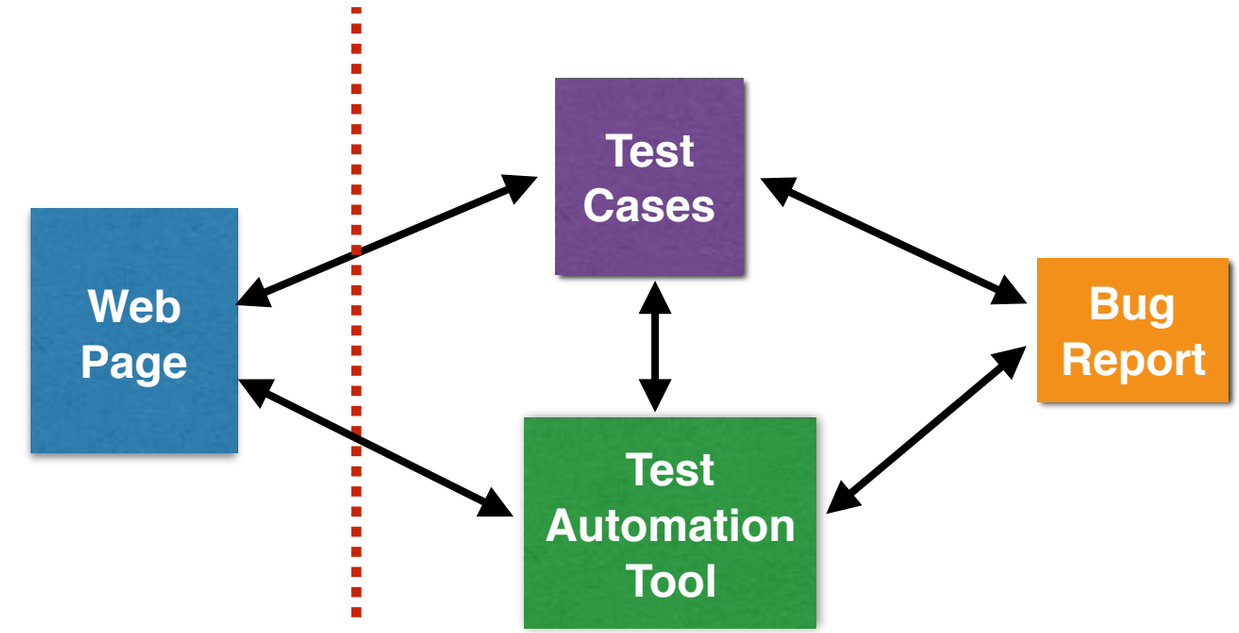
- Long N-tuples are used and given high weight, when available
- Short N-tuples are resorted to when longer N-tuples are unavailable

Open issues

- **Model inference**:
  - State abstraction function is heuristic and may be imprecise
  - Crawling may be incomplete
- **Test case generation**:
  - Input data generation is not considered
  - Computation of N-gram statistics may require substantial monitoring
  - Feasibility may involve both path selection and input data generation

Page Objects

An abstraction that exposes a model of a web page to test cases.

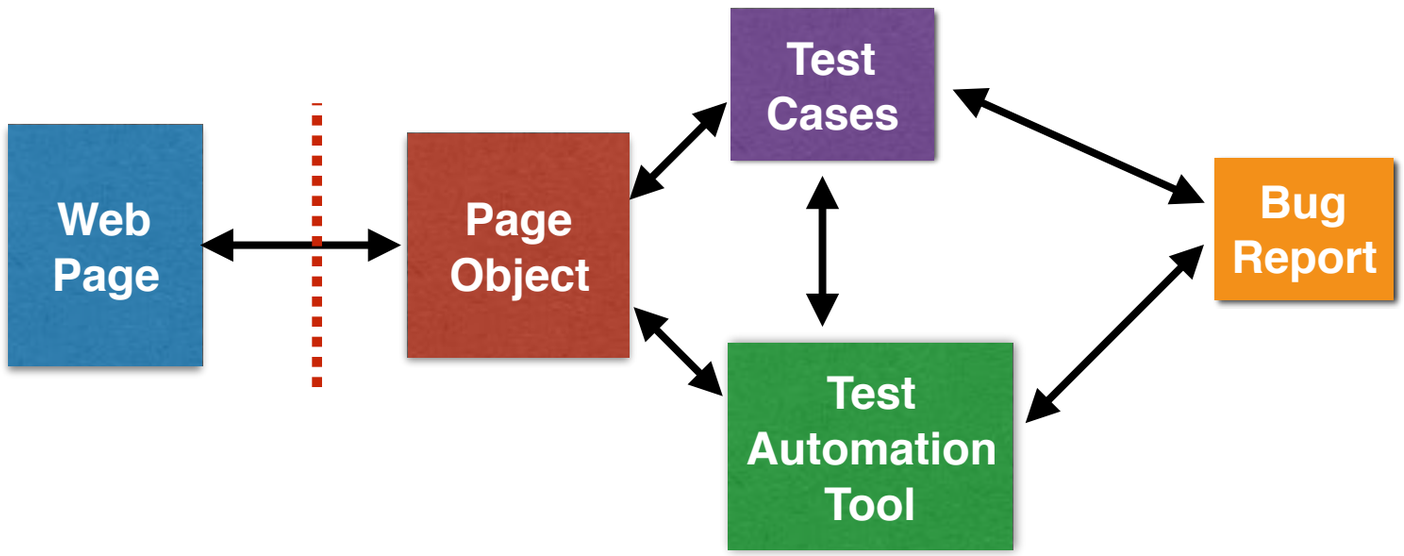



```
x = findElement(By.XPath(
  "//li[2]//[text()=
    'Ranking']"));
assert(x == "7");
```

HTML pages

Title: White
Artist: Red
Ranking: 1

Title: Blue
Artist: Grey
Ranking: 7




```
a = selectAlbum("Blue");
assert(a.getRanking() == 7);
```

Page Objects

```
class AlbumListPage {
  selectAlbum(String title)
```

```
class AlbumPage {
  getRanking()
```

Approach

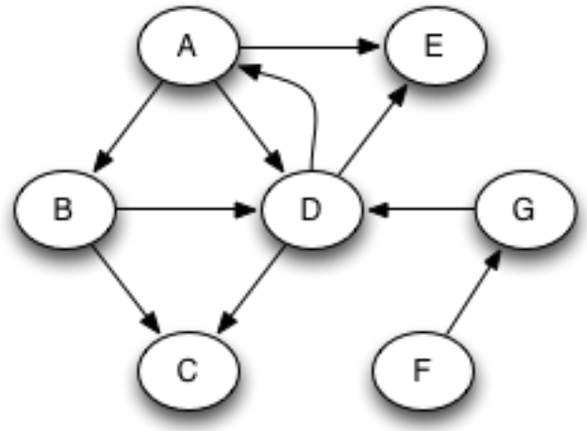
WEB APPLICATION

modeling

PAGE OBJECTS

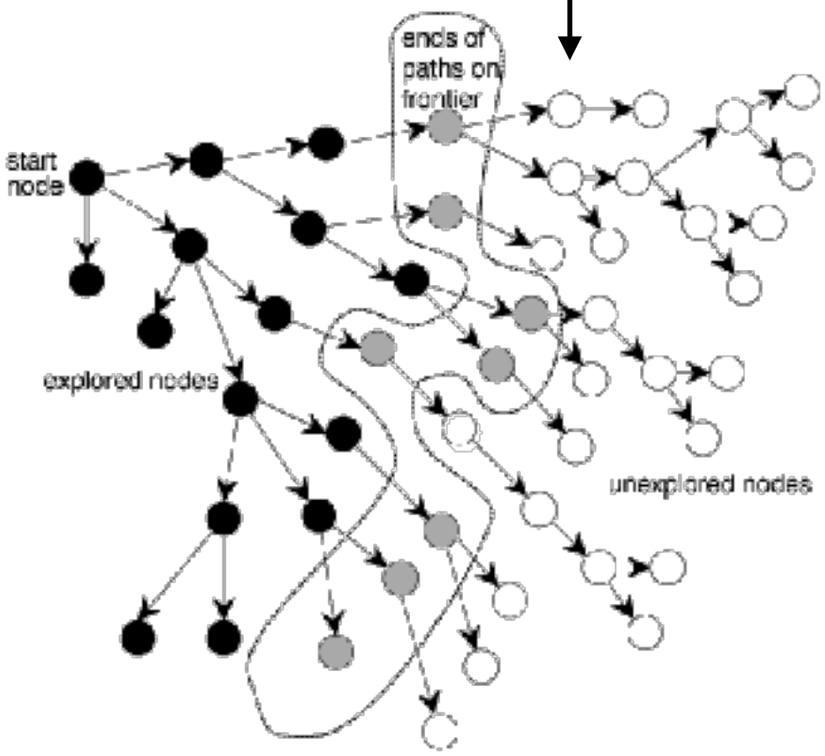
parsing

NAVIGATION GRAPH



state/transition coverage

PATH GENERATOR



from path to Java class

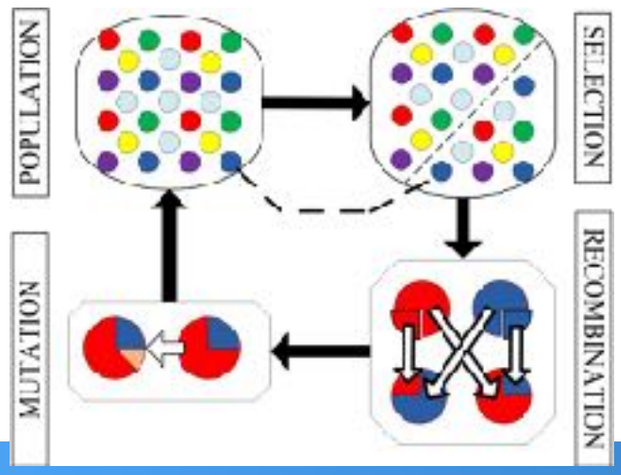
SEARCH BASED TEST CASE GENERATOR

EVOSUITE

Automatic Test Suite Generation for Java

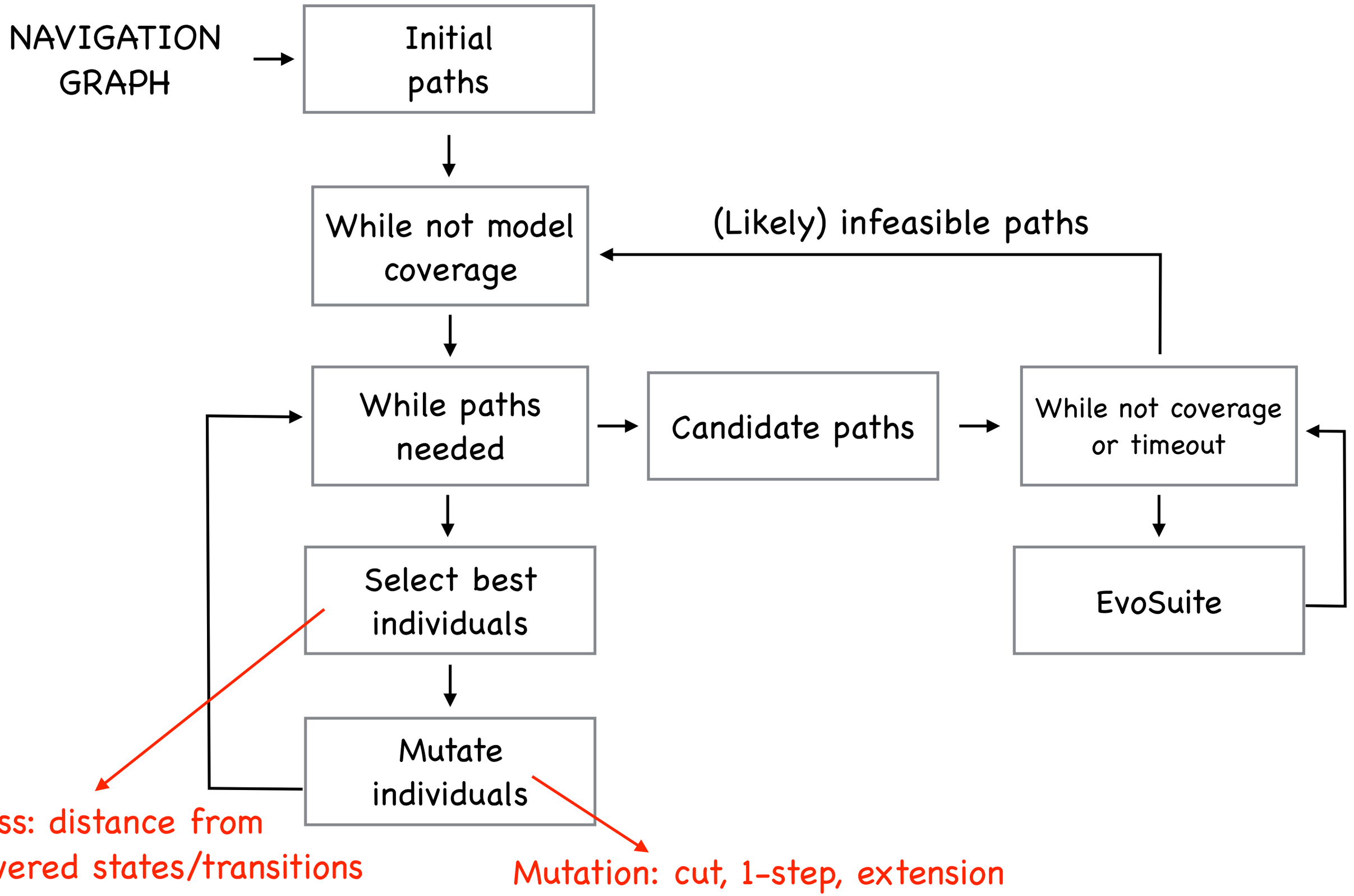
new paths

feasible
+
likely infeasible paths



SEARCH BASED PATH GENERATOR

Path and test case generators



Path generator

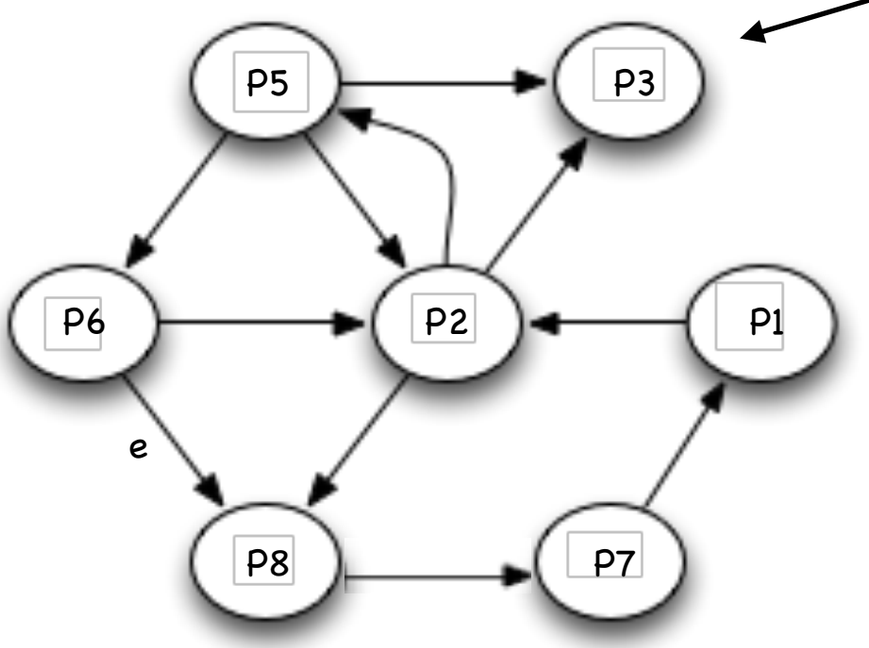
1. Cut: <P1, P2, P3> ····· p < 1 ·····▶ <P1, P2>

MUTATION OPERATORS

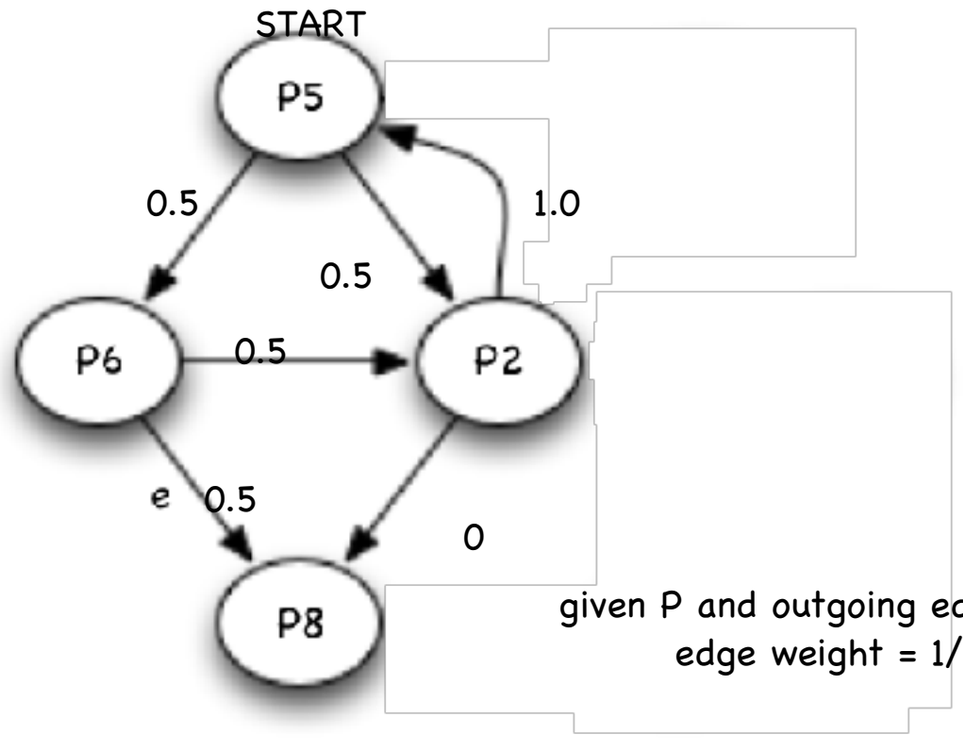
2. 1-step: <P1, P2> P2 → P3, P5

····· p < 1 ·····▶ <P1, P2, P5>

3. Extension: e = rand(X) <P1, P2, P5> ····· p = 1 ·····▶ <P1, P2, P5, P6, P8> ∉ set of likely unfeasible paths

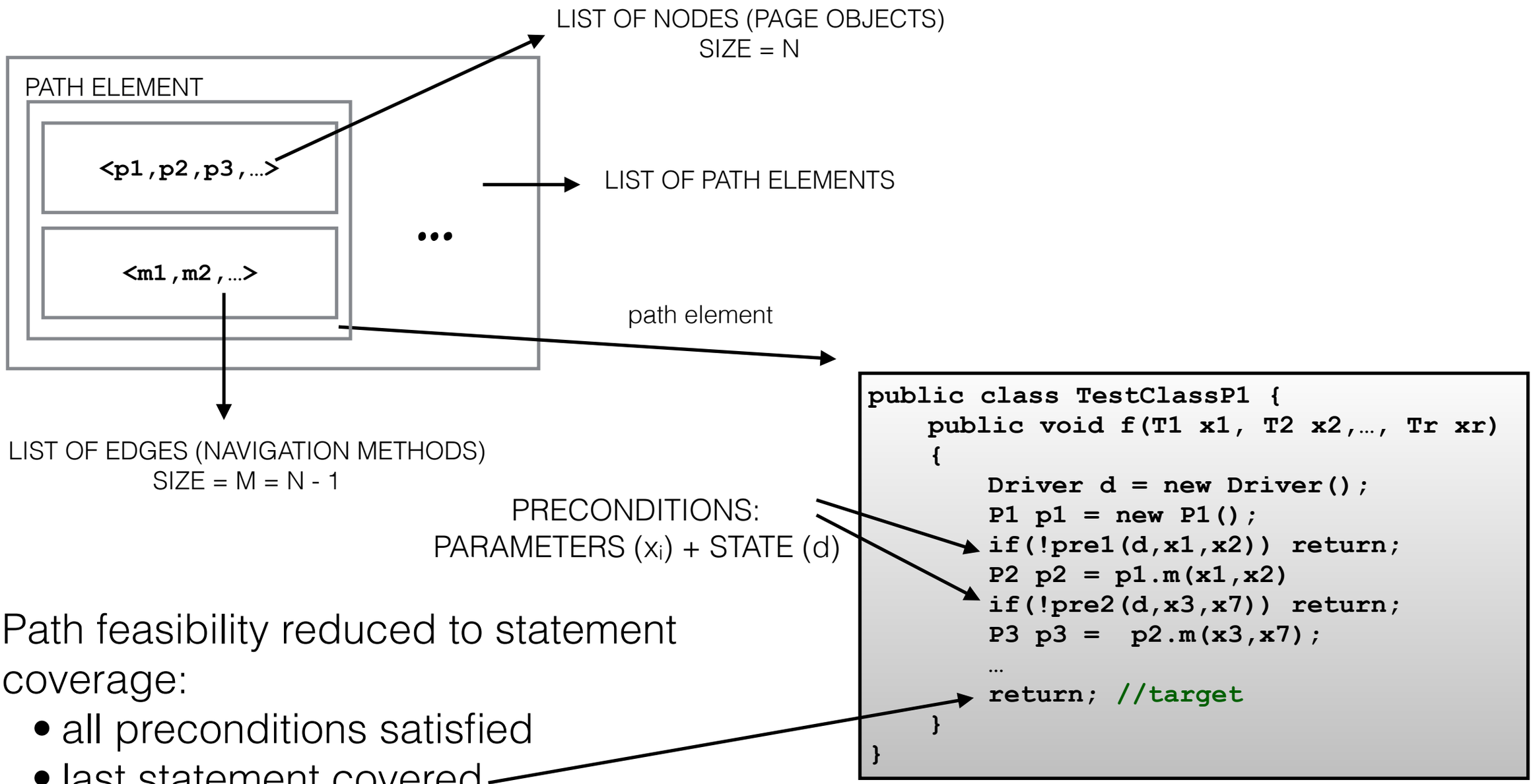


random walk



given P and outgoing edges (N):
edge weight = 1/N

Path feasibility



- Path feasibility reduced to statement coverage:
  - all preconditions satisfied
  - last statement covered

Relevance for GAUSS

- Models similar to Page Objects may be available in GAUSS
- Upon adaptation/evolution, models may include previously untested or scarcely tested behaviours
- Path generation and path feasibility is key for automated testing of new behaviours
- Availability of automated oracle is another key prerequisite