ECE570 Lecture 9: Nondeterminism

Jeffrey Mark Siskind

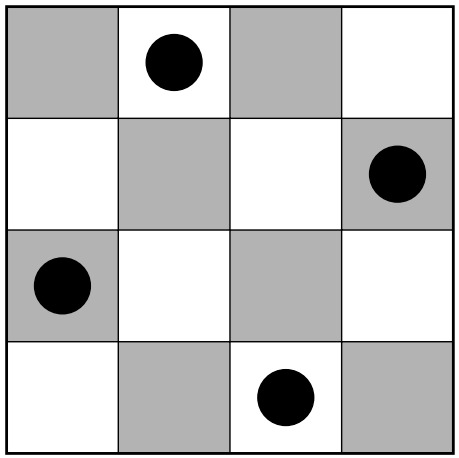
School of Electrical and Computer Engineering

Fall 2013

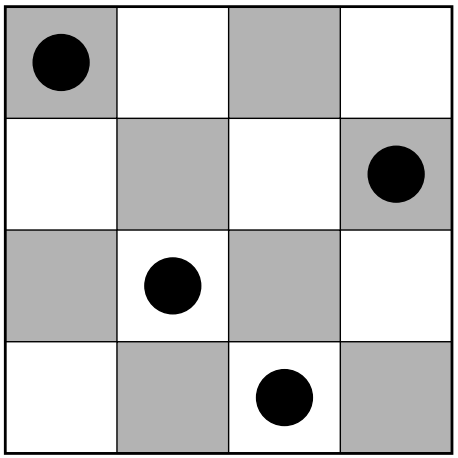
PURDUE
UNIVERSITY

N Queens

Problem: Place N Queens on an $N \times N$ chess-board so that they don't attack one another.



Solution



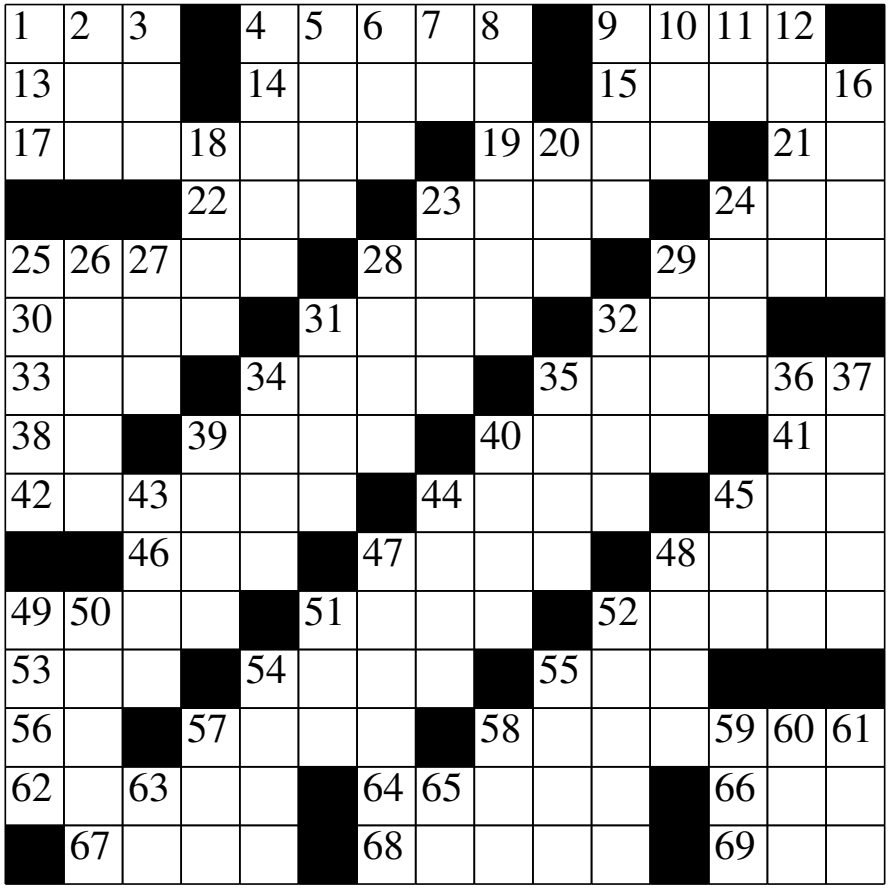
Non-Solution

Crossword Puzzles—I

Problem: Fill a crossword puzzle with words from a lexicon so that intersecting words have the same letter at their intersection.

ad	al	alas	aloha	art	at	atl	bags	bang
base	bore	coat	dad	dart	dime	dine	dive	do
eh	elf	er	evade	even	fan	fee	fine	gate
goat	happy	hares	hem	hide	hire	hive	hoe	hone
inn	largest	learned	lee	lemons	lid	lilac	lip	lo
load	mates	mile	mirror	mist	moon	more	oak	olive
ore	pans	paris	pay	pea	pedal	penny	pier	pile
pins	pits	raise	rips	roe	ropes	roy	salads	see
slam	slat	some	spot	steer	stew	tag	tame	tan
tank	tea	tee	tie	tigers	tire	to	toe	wager
wave	wider	win	wires					

Crossword Puzzles—II



Constraint Satisfaction Problems—I

A set of variables: $x_1, \ldots, x_n$

A set of domains: $D_1, \ldots, D_n$

A set of constraints: $P_j(x_{i_1}, \ldots, x_{i_m})$

Problem: Find $x_1 \in D_1, \ldots, x_n \in D_n$ such that all constraints are met.

Variants:

satisfiability
: Does there exist a solution?

solution
: Find one solution.
Find all solutions.
Find the best solution.
Count the solutions.

Constraint Satisfaction Problems—II

Special Cases:

Boolean CSP: domains are $\{\textbf{true}, \textbf{false}\}$

Binary CSP: constraints all have two arguments

How are Constraints Specified?

- Computable Predicates
- Tables
  - list all tuples
  - list only true tuples
  - list only false tuples

N Queens as a CSP—Naive Encoding—I

Variables: $q_{i,j}$ for $1 \leq i,j \leq n$

Domains: $D_{ij} = \{\textbf{true}(1), \textbf{false}(0)\}$

Constraints:

$$attack(i_1,j_1,i_2,j_2) \leftrightarrow \begin{array}{l}(i_1 = i_2 \vee j_1 = j_2 \vee |i_1 - i_2| = |j_1 - j_2|) \wedge \\ (i_1 \neq i_2 \vee j_1 \neq j_2)\end{array}$$

N Queens as a CSP—Naive Encoding—II

$$P_1(q_{i_1,j_1}, q_{i_2,j_2}) \leftrightarrow \overline{q_{i_1,j_1} \wedge q_{i_2,j_2}} \quad \text{where} \quad attack(i_1, j_1, i_2, j_2)$$

$$\left(\sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} \right) = n$$

$$P_2(q_{11}, \ldots, q_{1n}, \ldots, q_{n1}, \ldots, q_{nn})$$

While P_1 is binary, P_2 has n^2 arguments!

N Queens as a CSP—A Better Encoding

Observations: A row can have at most one queen.
A row must have at least one queen.

Variables: q_i for $1 \leq i \leq n$

Domains: $D_i = \{1, \ldots, n\}$

Constraints: $P(q_i, q_j) \leftrightarrow \overline{attack(i, q_i, j, q_j)}$ where $i \neq j$

Now, the CSP is binary.

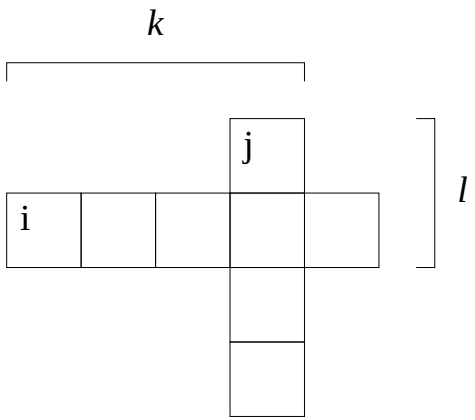
Lesson: Formulate CSP using knowledge of problem structure.

Crossword Puzzles as CSPs

Variables: $across_i, down_j$

Domains: words in lexicon of appropriate length

Constraints:



$$P(across_i, down_j) \leftrightarrow across_i[k] = down_j[l]$$

Solving CSPs: Generate & Test

for $x_1 \in D_1$
do for $x_2 \in D_2$
do
$\vdots$
for $x_n \in D_n$
do if all constraints are satisfied
then $\langle x_1, \ldots, x_n \rangle$ is a solution **fi od** ... **od od**

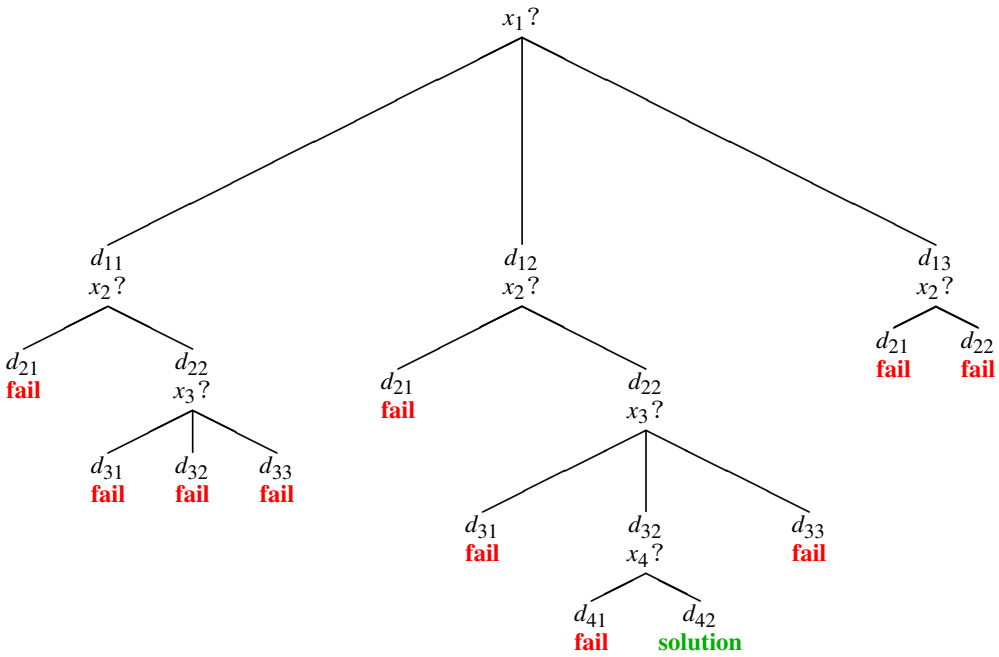
Solving CSPs: Early Failure Detection

for $x_1 \in D_1$
do if constraints over x_1 are satisfied
  then for $x_2 \in D_2$
    do if constraints over x_1 and x_2 are satisfied
      then

$$\vdots$$

for $x_n \in D_n$
do if all constraints are satisfied
then $\langle x_1, \ldots, x_n \rangle$ is a solution
fi od ... **fi od fi od**

Search Trees—I



Search Trees—II

non-leaf nodes correspond to variables
leaf nodes are labeled either **solution** or **failure**
arcs are labeled with choices

Variable order: fixed or dynamic
induces a search tree

Search order: breadth-first
depth-first left-to-right

Early Failure Detection: full vs. sparse tree
inference applied at nodes

Nondeterministic Scheme

`(+ 3 4)` $\Longrightarrow$ `7`

`(either 1 2)` $\Longrightarrow$ $\{$`1 2`$\}$

`(either (either 1 2) (either 2 3))` $\Longrightarrow$ $\{$`1 2 2 3`$\}$

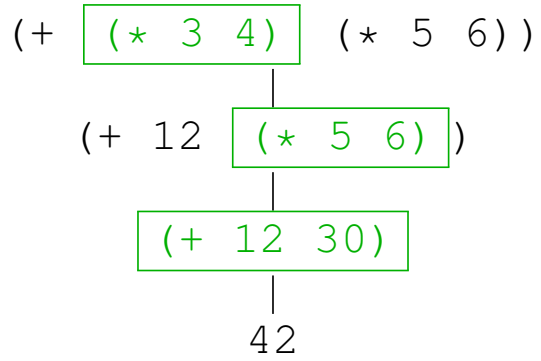
`(+ (either 1 2) 1)` $\Longrightarrow$ $\{$`2 3`$\}$

`(+ (either 1 2) (either 2 3))` $\Longrightarrow$ $\{$`3 4 4 5`$\}$

`(fail)` $\Longrightarrow$ $\{\}$

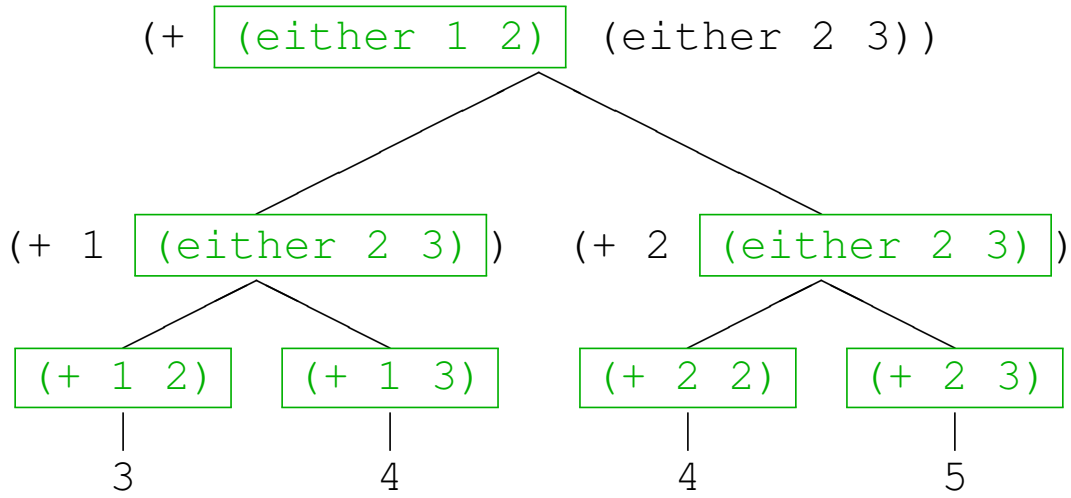
`either` sometimes called `amb` (McCarthy, 1963)
or `choose` (Chapman, 1985)

Search-Tree Semantics—I



- single linear reduction modulo execution order
- applicative order

Search-Tree Semantics—II



- single reduction tree modulo execution order
- applicative order
- depth-first left-to-right search

BEGIN Syntax

$$\left.\begin{array}{l}\texttt{(begin } e_1 \\ \quad\quad e_2 \\ \quad\quad \vdots \\ \quad\quad e_{n-1} \\ \quad\quad e_n\texttt{)}\end{array}\right\} \rightsquigarrow \left\{\begin{array}{l}\texttt{(let* ((ignore } e_1\texttt{)} \\ \quad\quad\quad\quad \texttt{(ignore } e_2\texttt{)} \\ \quad\quad\quad\quad \vdots \\ \quad\quad\quad\quad \texttt{(ignore } e_{n-1}\texttt{))} \\ \quad e_n\texttt{)}\end{array}\right.$$

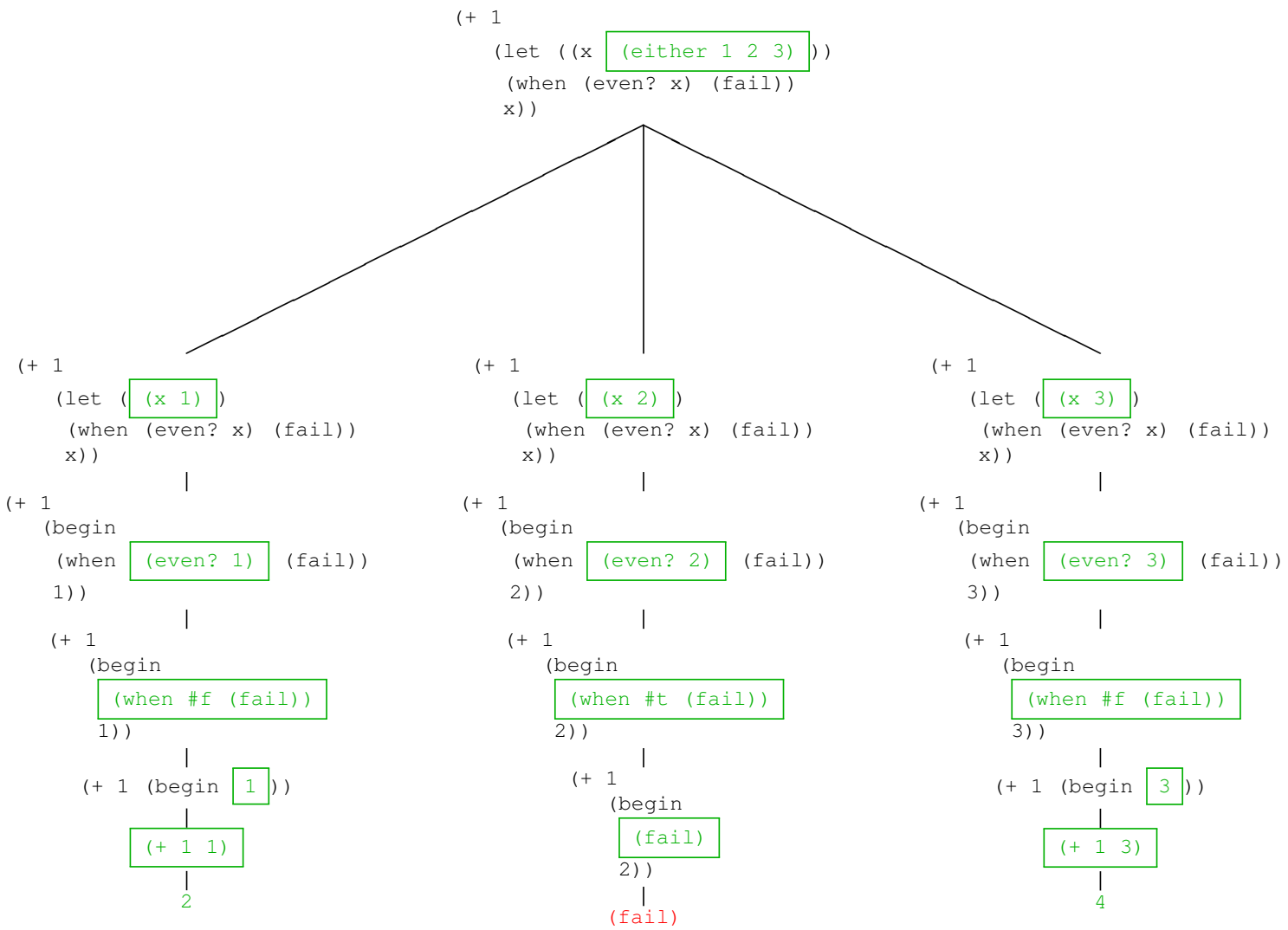
WHEN Syntax

$$\left.\begin{array}{l}\texttt{(when } e \\ \quad e_1 \\ \quad \vdots \\ \quad e_n\texttt{)}\end{array}\right\} \rightsquigarrow \left\{\begin{array}{l}\texttt{(if } e \\ \qquad \texttt{(begin } e_1 \\ \qquad\qquad \vdots \\ \qquad\qquad e_n\texttt{))}\end{array}\right.$$

UNLESS Syntax

$$\left.\begin{array}{l}\texttt{(unless } e \\ e_1 \\ \vdots \\ e_n\texttt{)}\end{array}\right\} \rightsquigarrow \left\{\begin{array}{l}\texttt{(if (not } e\texttt{)} \\ \quad\texttt{(begin } e_1 \\ \qquad\qquad \vdots \\ \qquad\qquad e_n\texttt{))}\end{array}\right.$$

Failure


(+ 1
(let ((x (either 1 2 3)))
(when (even? x) (fail))
x))
(+ 1
(let ((x (x 1)))
(when (even? x) (fail))
x))
(+ 1
(begin
(when (even? 1) (fail))
1))
(+ 1
(begin
(when #f (fail))
1))
(+ 1 (begin 1))
(+ 1 1)
2
(+ 1
(let ((x (x 2)))
(when (even? x) (fail))
x))
(+ 1
(begin
(when (even? 2) (fail))
2))
(+ 1
(begin
(when #t (fail))
2))
(+ 1
(begin
(fail)
2))
(fail)
(+ 1
(let ((x (x 3)))
(when (even? x) (fail))
x))
(+ 1
(begin
(when (even? 3) (fail))
3))
(+ 1
(begin
(when #f (fail))
3))
(+ 1 (begin 3))
(+ 1 3)
4

Combinatorial Enumeration

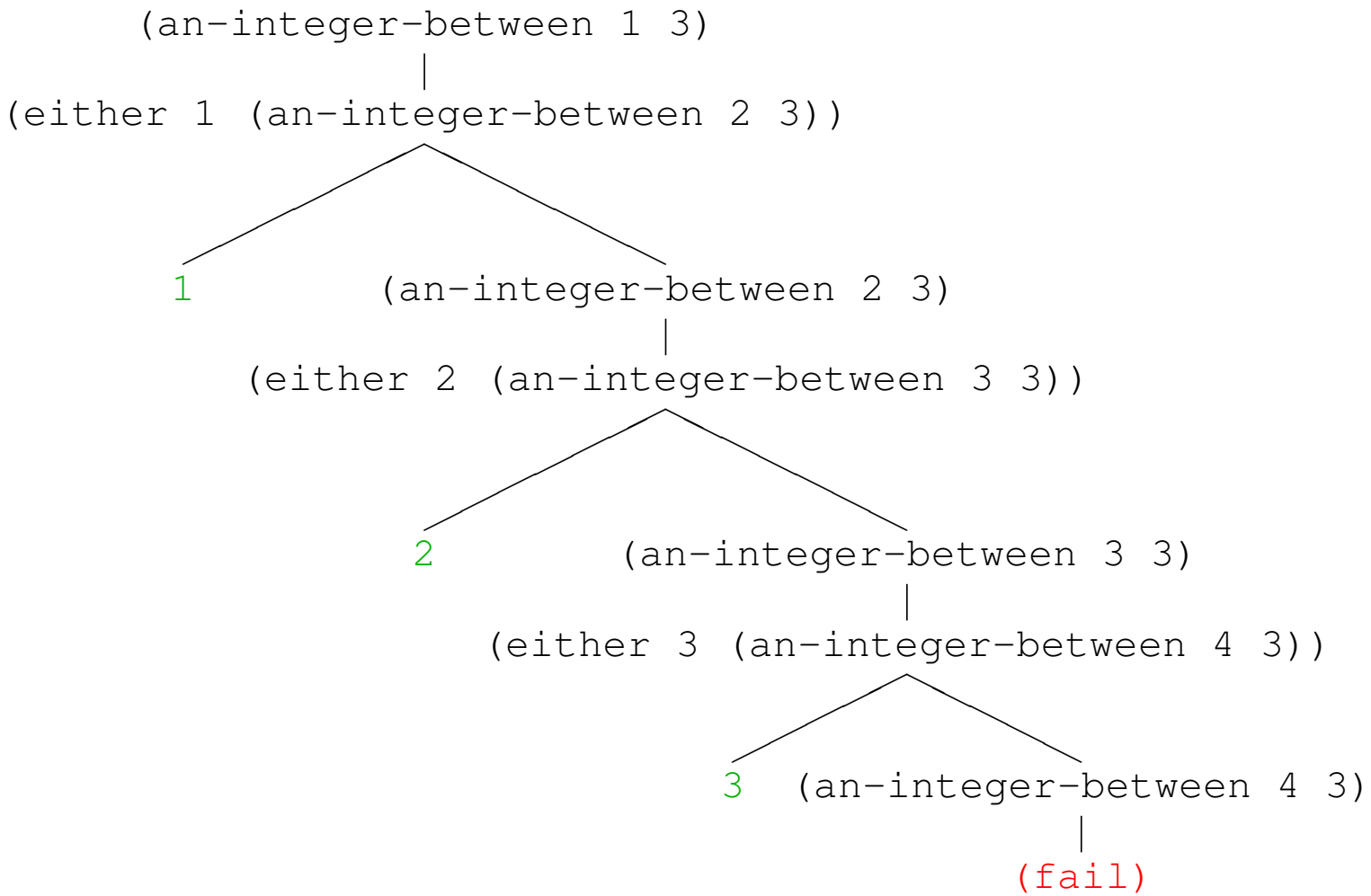
```
(define (an-integer-between low high)
 (when (> low high) (fail))
 (either low (an-integer-between (+ low 1) high)))
```

`(an-integer-between 1 5)` $\Longrightarrow$ {`1 2 3 4 5`}

`(an-integer-between 10 5)` $\Longrightarrow$ {}

- `an-integer-between` is defined in QOBISCHEME.

Induced Search Tree

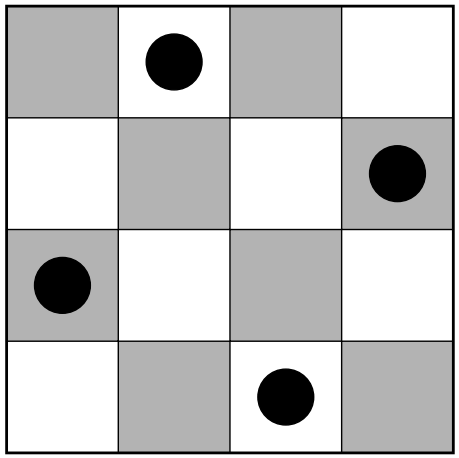


Pythagorean Triples

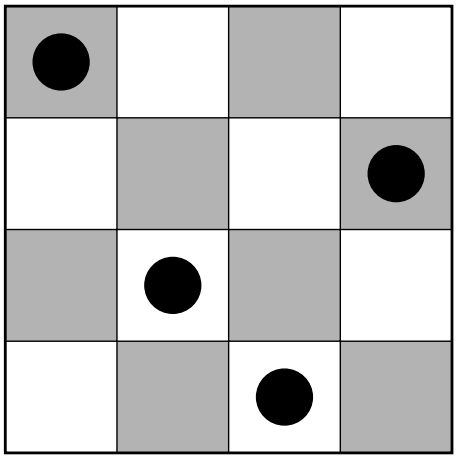
```
(define (a-pythagorean-triple n)
 (let ((a (an-integer-between 1 n))
       (b (an-integer-between 1 n))
       (c (an-integer-between 1 n)))
  (unless (= (+ (* a a) (* b b)) (* c c)) (fail))
  (list a b c)))
```

`(a-pythagorean-triple 10)` $\Longrightarrow$ {`(3 4 5)` `(4 3 5)` `(6 8 10)` `(8 6 10)`}

N Queens (Generate & Test)—I



Solution



Non-Solution

```
(define (attacks? qi qj delta-rows)
 (or (= qi qj) (= (abs (- qi qj)) delta-rows)))

(define (list-of n f)
 (if (= n 0) '() (cons (f) (list-of (- n 1) f))))
```

Looping

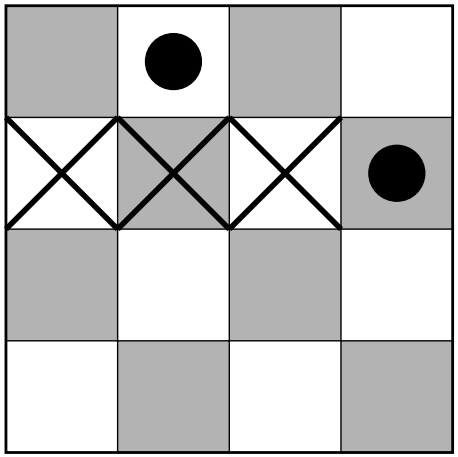
```
(define (for-each p l)
 (unless (null? l)
  (p (first l))
  (for-each p (rest l))))

(define (for-each-indexed p l)
 (define (loop l i)
  (unless (null? l)
   (p (first l) i)
   (loop (rest l) (+ i 1))))
 (loop l 0))
```

N Queens (Generate & Test)—II

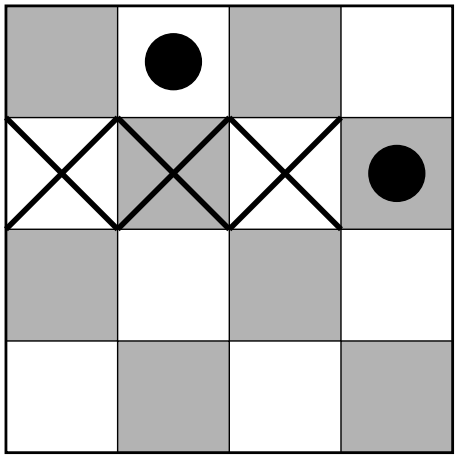
```
(define (n-queens n)
 (let ((queens (list-of
                n
                (lambda () (an-integer-between 0 (- n 1))))))
  (for-each-indexed
   (lambda (qi i)
    (for-each-indexed
     (lambda (qj j)
      (when (attacks? qi qj (abs (- i j))) (fail)))
     queens))
   queens)
  queens))
```

N Queens (Early Failure Detection)—I



```
(define (check-queens new-column old-columns)
 (for-each-indexed
  (lambda (old-column i)
   (when (attacks? new-column old-column (+ i 1))
    (fail)))
  old-columns))
```

N Queens (Early Failure Detection)—II



```
(define (n-queens n)
 (define (loop columns)
  (if (= (length columns) n)
      columns
      (let ((column (an-integer-between 0 (- n 1))))
       (check-queens column columns)
       (loop (cons column columns)))))
 (loop '()))
```

The Future

Today we have formulated the following problems as CSPs:

- N Queens
- Crossword Puzzles

Over the course of the rest of the semester we will formulate the following additional problems as CSPs:

- temporal reasoning
- scene labeling
- qualitative physical reasoning
- multiple-fault diagnosis
- planning