Individual Project Thoughts 1

Functional Dependencies (aka "Determinations" in AI)

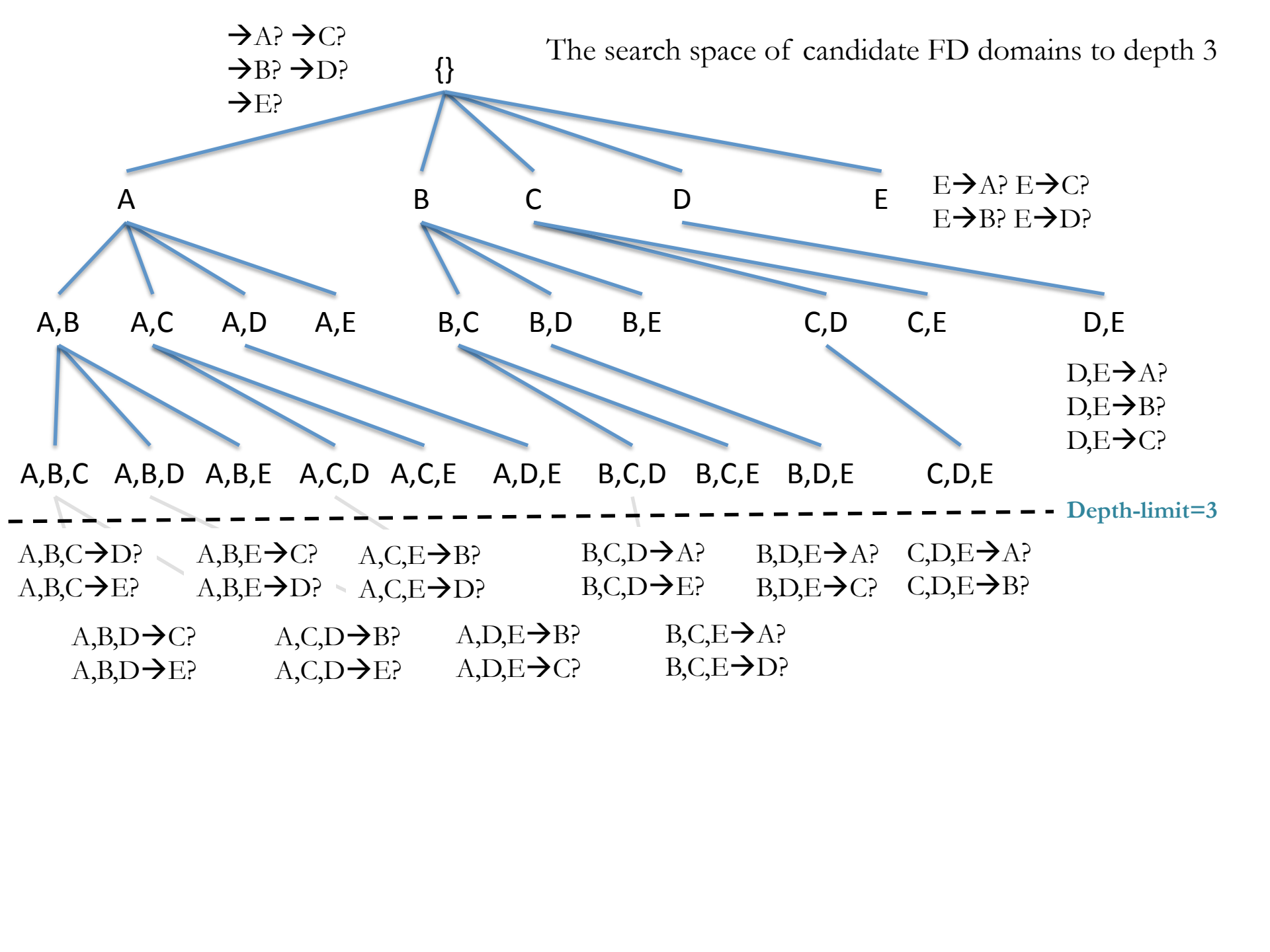
Suppose we have a universal relation with attributes A, B, C, …, each with a set of possible values (e.g., attribute A can have values a1, a2, a3, …ai)

A	B	C	D	E	F	G	H …
a1	b3	c2	d5	e7	f3	g1	h6 …
a4	b2	c2	d4	e2	f1	g1	h5 …
a2	b1	c1	d2	e5	f5	g3	h2 …
a1	b3	c3	d5	e6	f4	g1	h8 …
…							

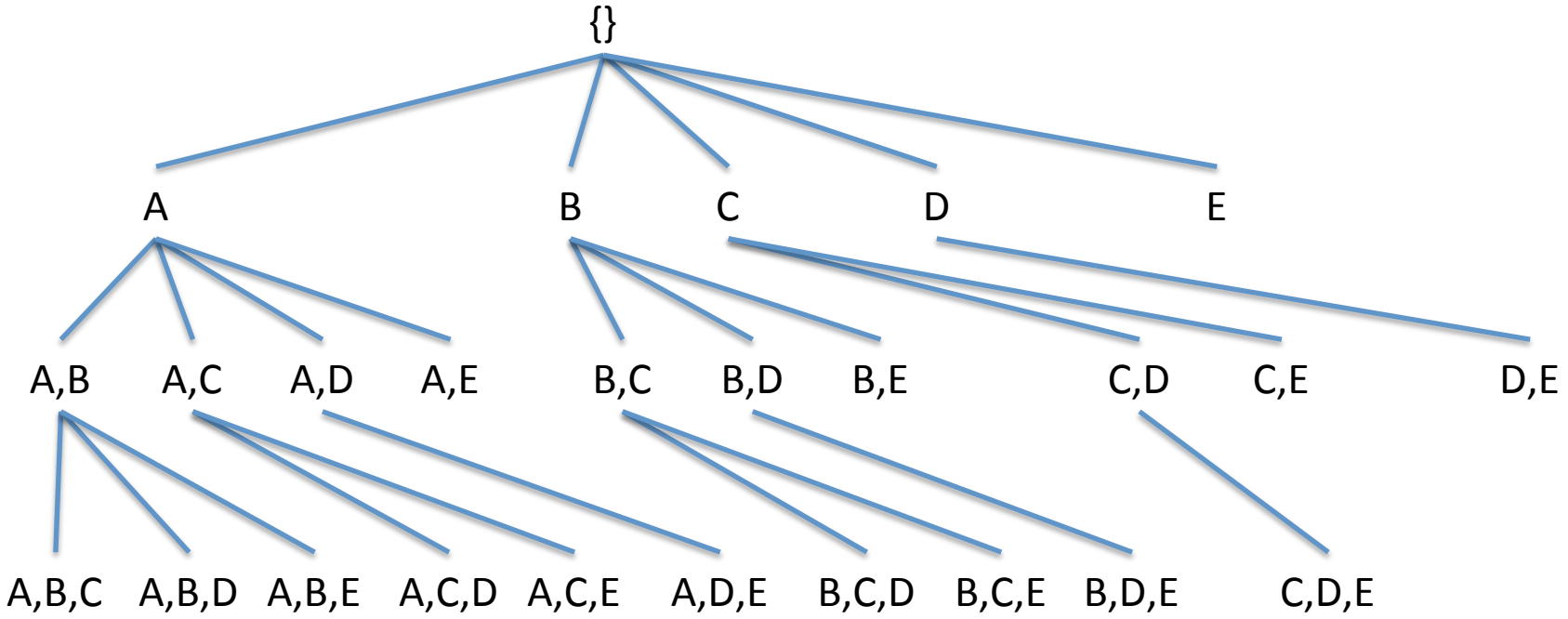
Suppose we are not told the FDs that are manifest (or intended to be manifest) in this universal relation

How can we induce the FDs through a process of "unsupervised" machine learning?

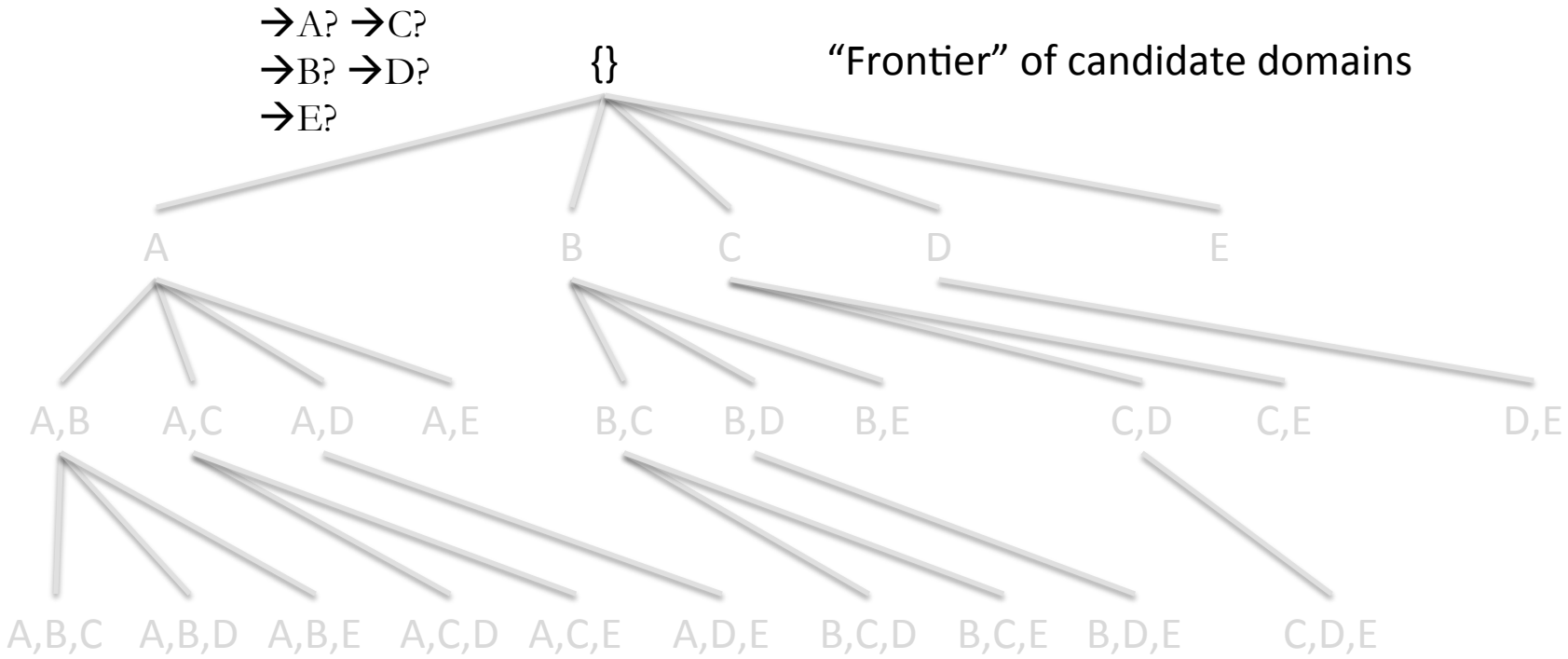
Schlimmer, J. (1993). Efficiently Inducing Determinations: A Complete and Systematic Search Algorithm that Uses Optimal Pruning (1993)
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.2038


→A? →C?
→B? →D?
→E?
{}
The search space of candidate FD domains to depth 3
A
B
C
D
E
E→A? E→C?
E→B? E→D?
A,B
A,C
A,D
A,E
B,C
B,D
B,E
C,D
C,E
D,E
D,E→A?
D,E→B?
D,E→C?
A,B,C
A,B,D
A,B,E
A,C,D
A,C,E
A,D,E
B,C,D
B,C,E
B,D,E
C,D,E
Depth-limit=3
A,B,C→D?
A,B,C→E?
A,B,E→C?
A,B,E→D?
A,C,E→B?
A,C,E→D?
B,C,D→A?
B,C,D→E?
B,D,E→A?
B,D,E→C?
C,D,E→A?
C,D,E→B?
A,B,D→C?
A,B,D→E?
A,C,D→B?
A,C,D→E?
A,D,E→B?
A,D,E→C?
B,C,E→A?
B,C,E→D?

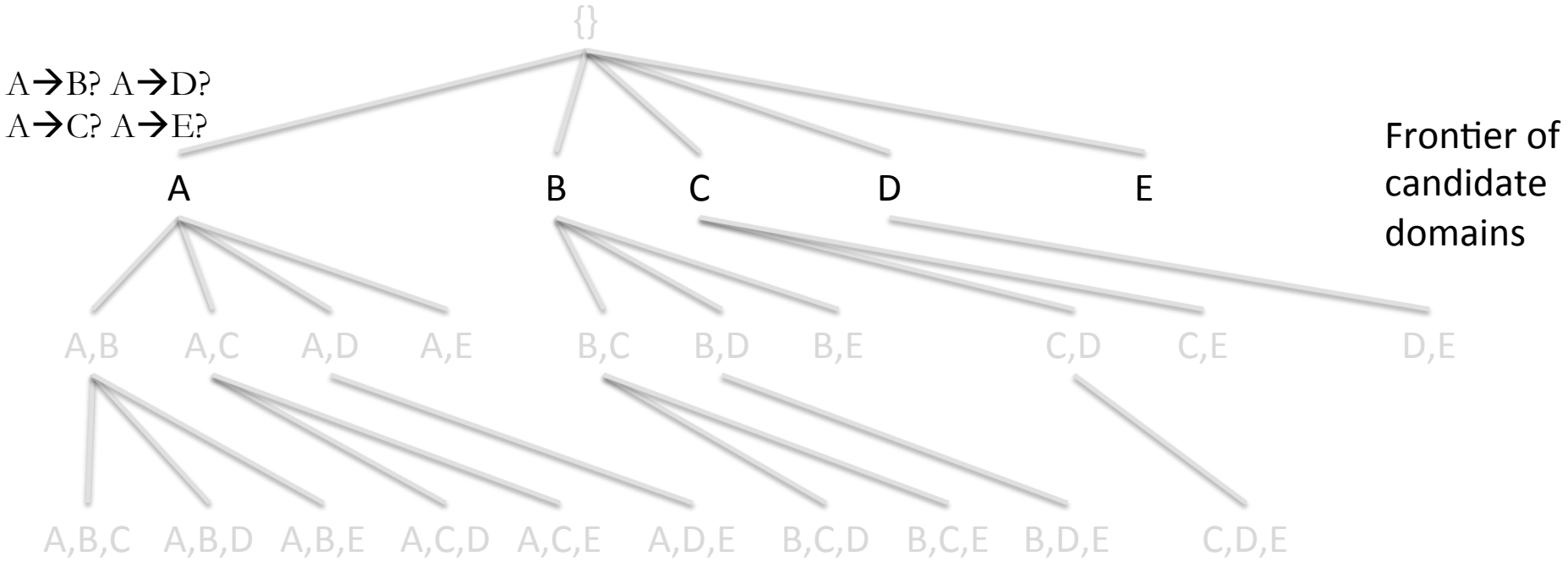
This space can be searched by breadth-first search, depth-first search, or a more sophisticated heuristic search



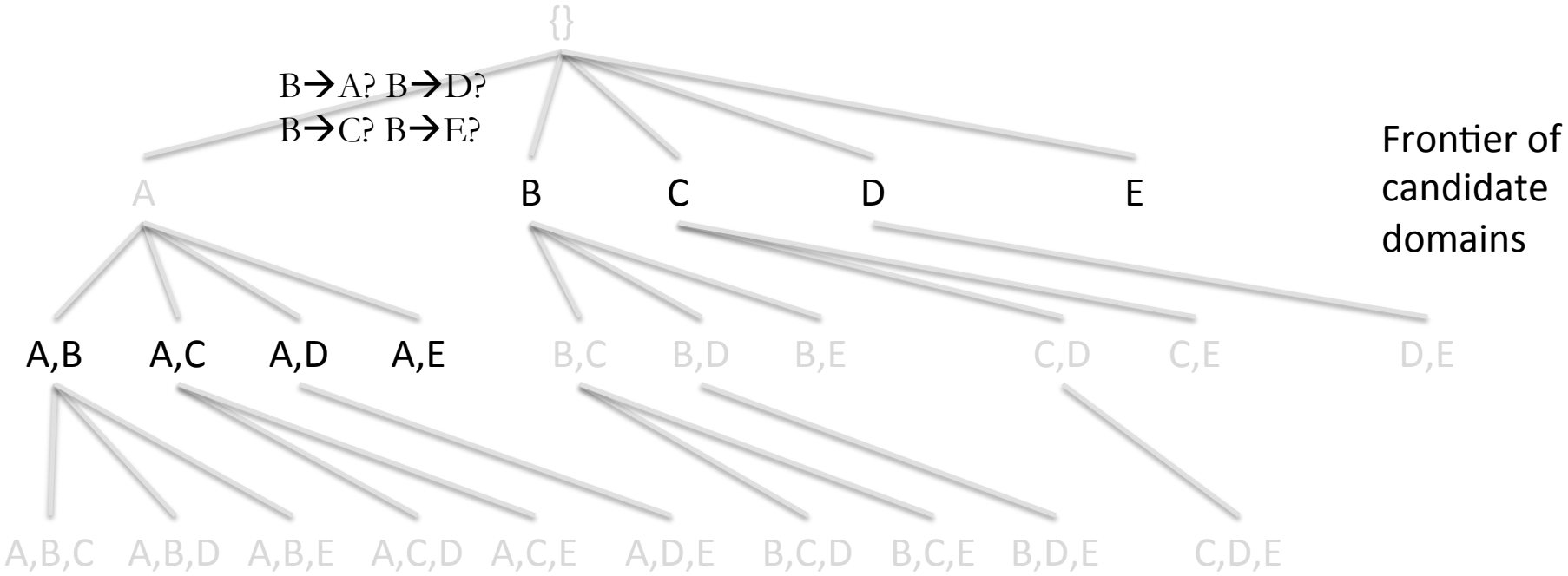
Breadth first search



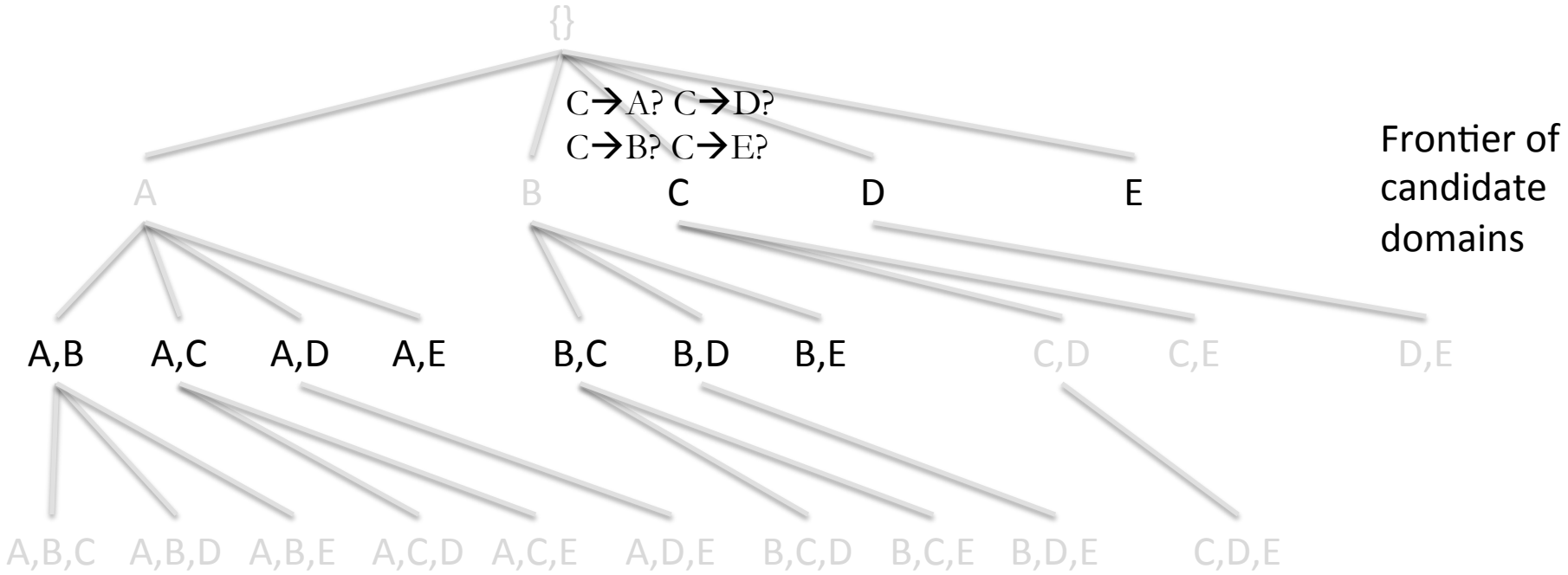
Breadth first search



Breadth first search

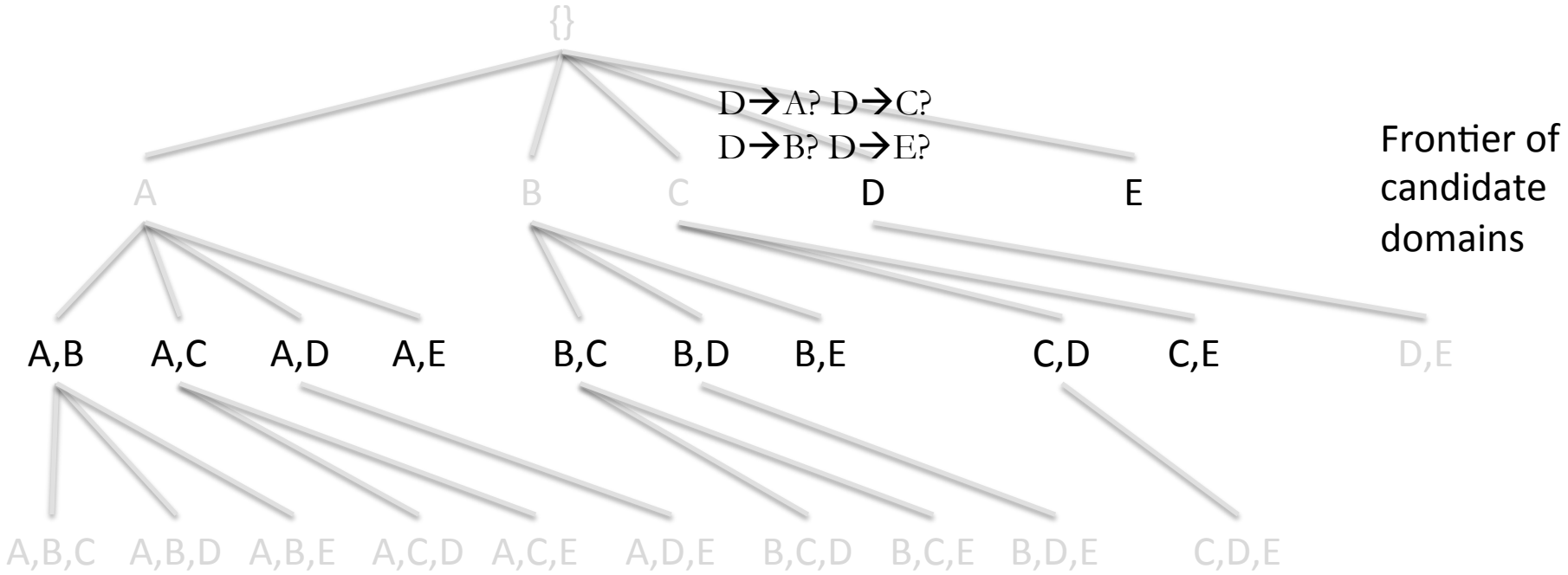


Breadth first search

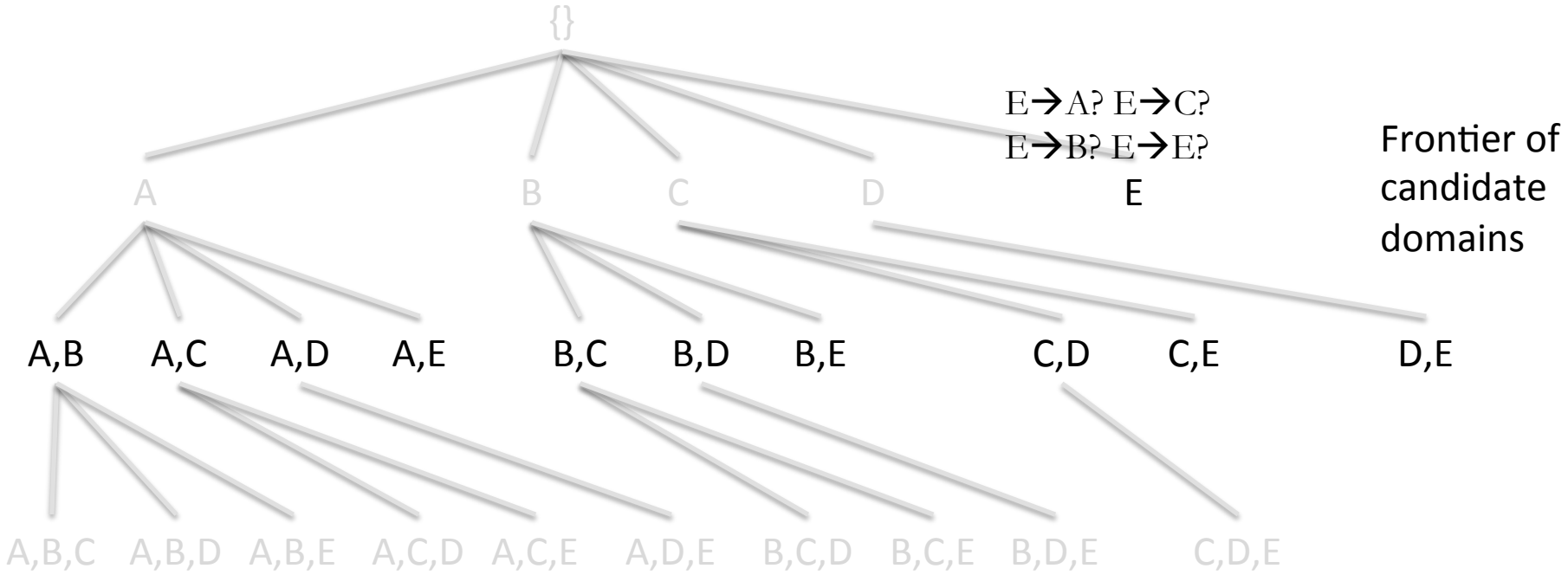


Frontier of candidate domains

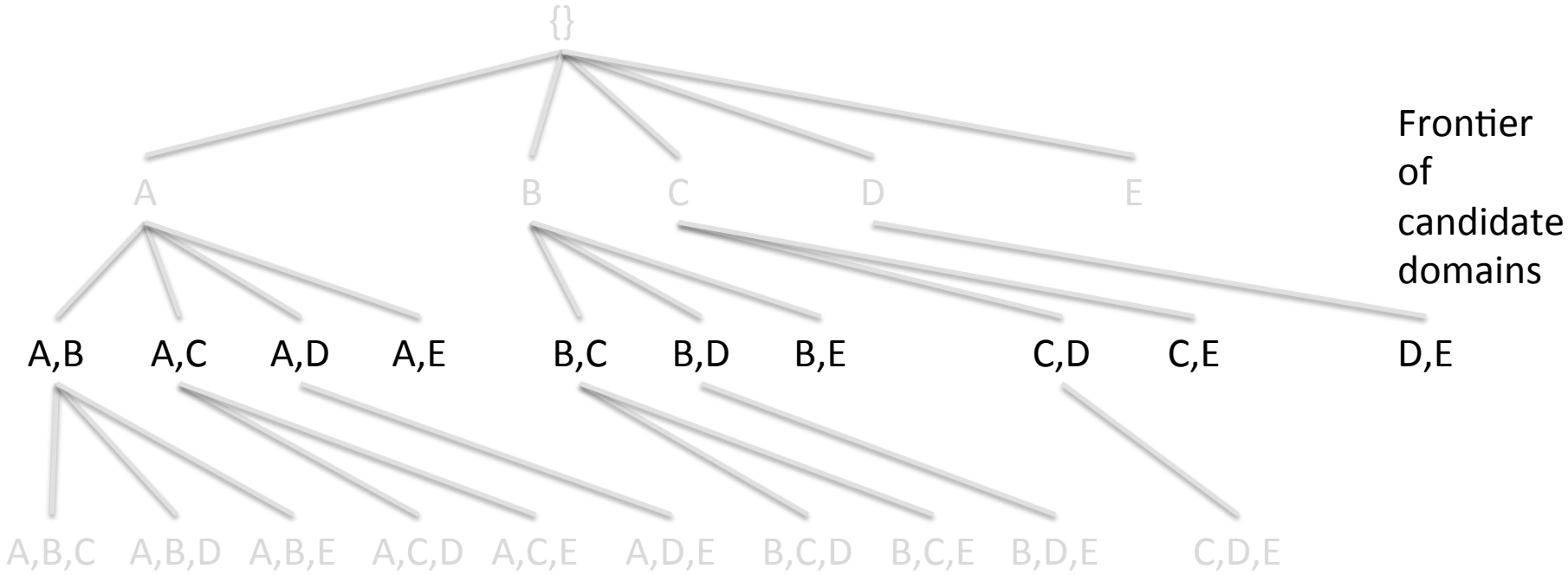
Breadth first search



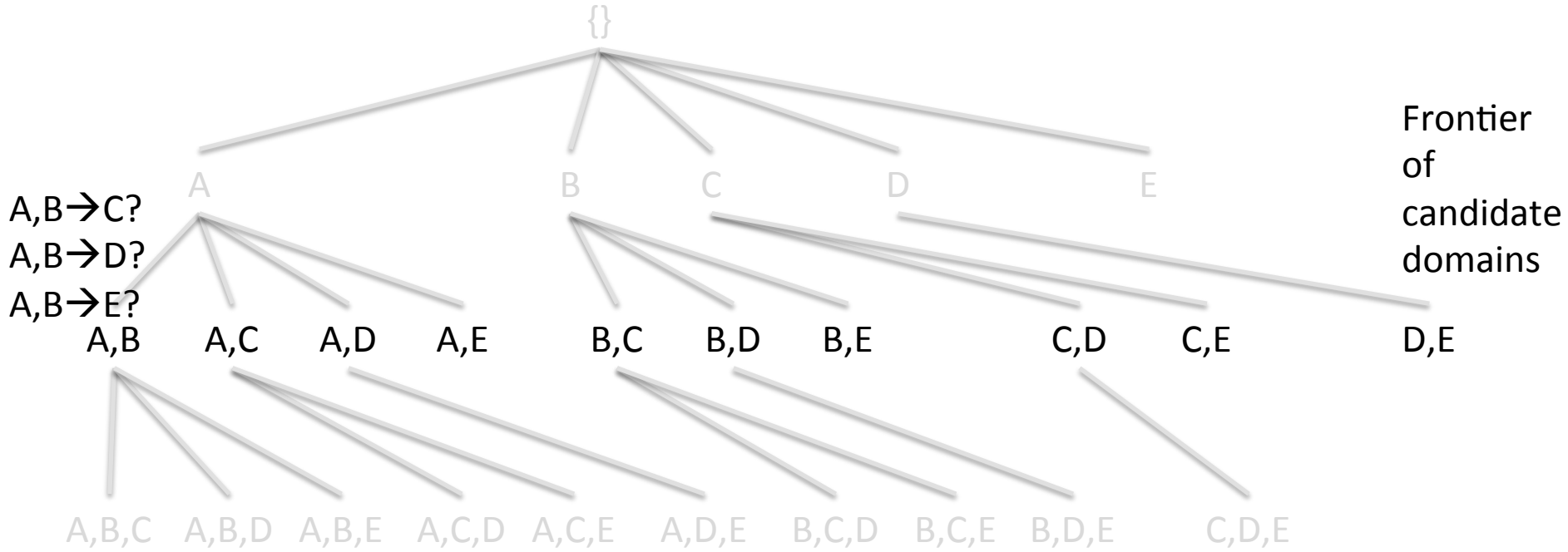
Breadth first search



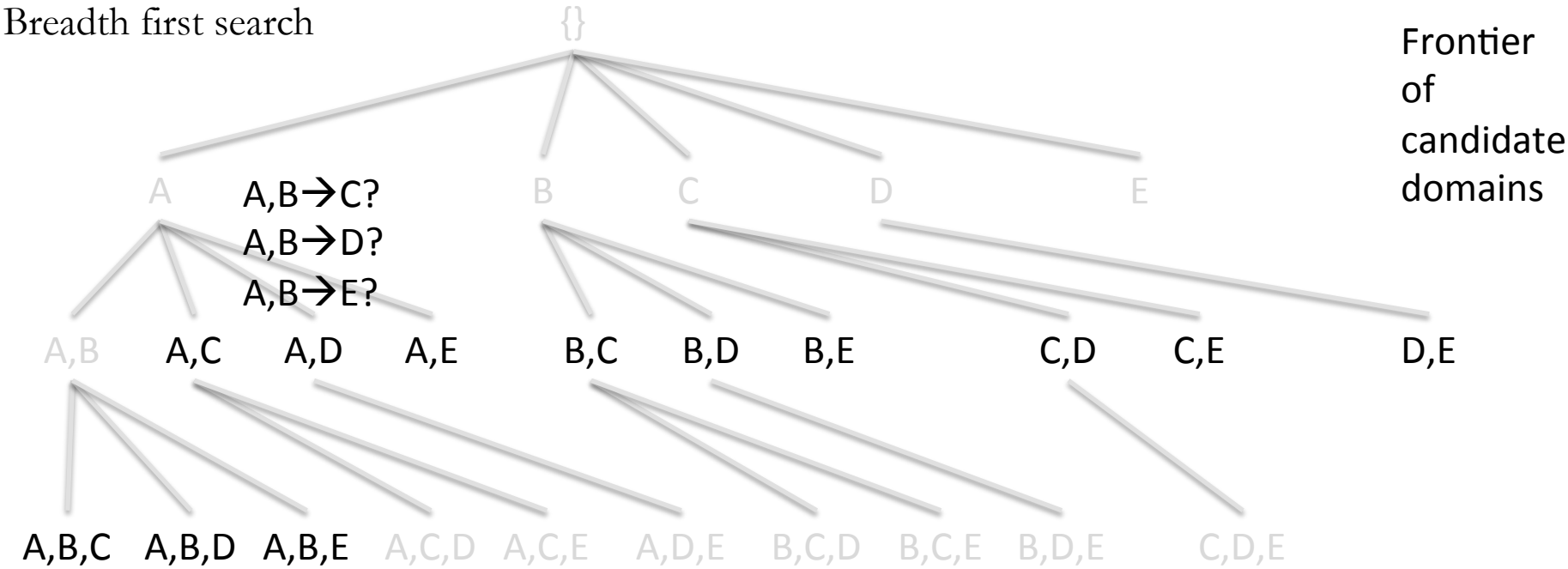
Breadth first search



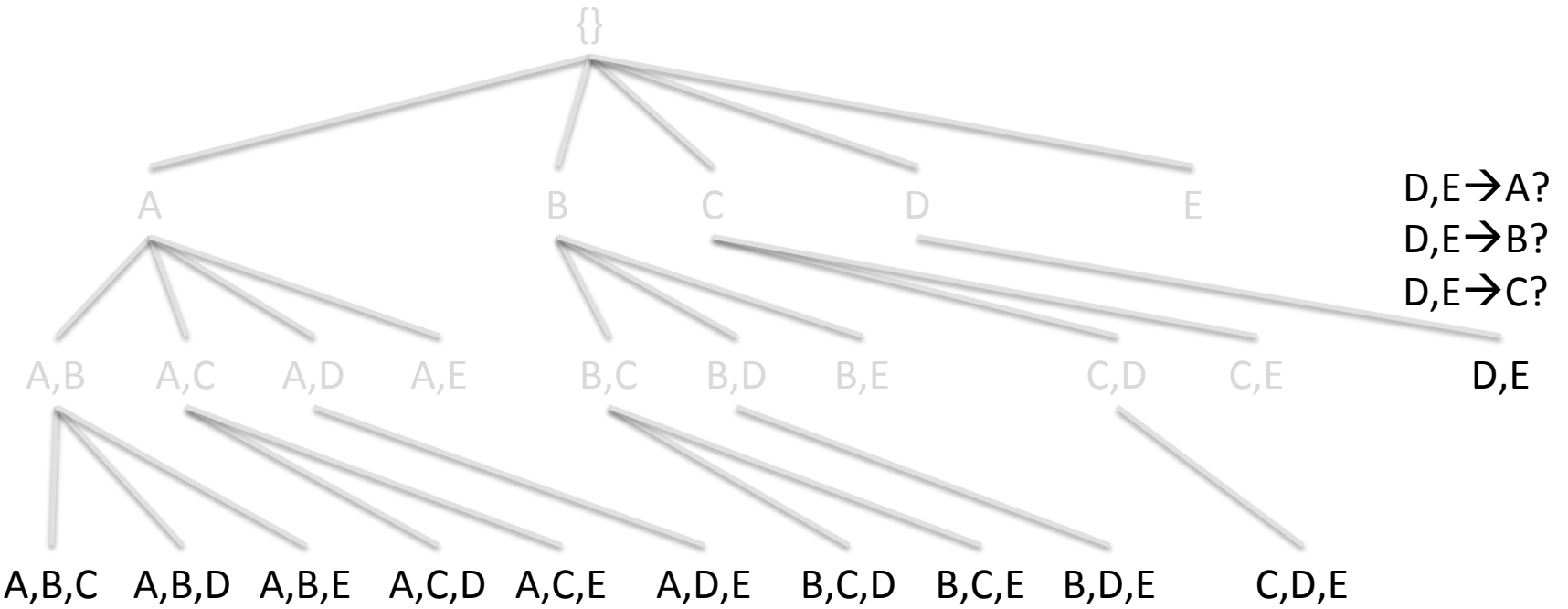
Breadth first search

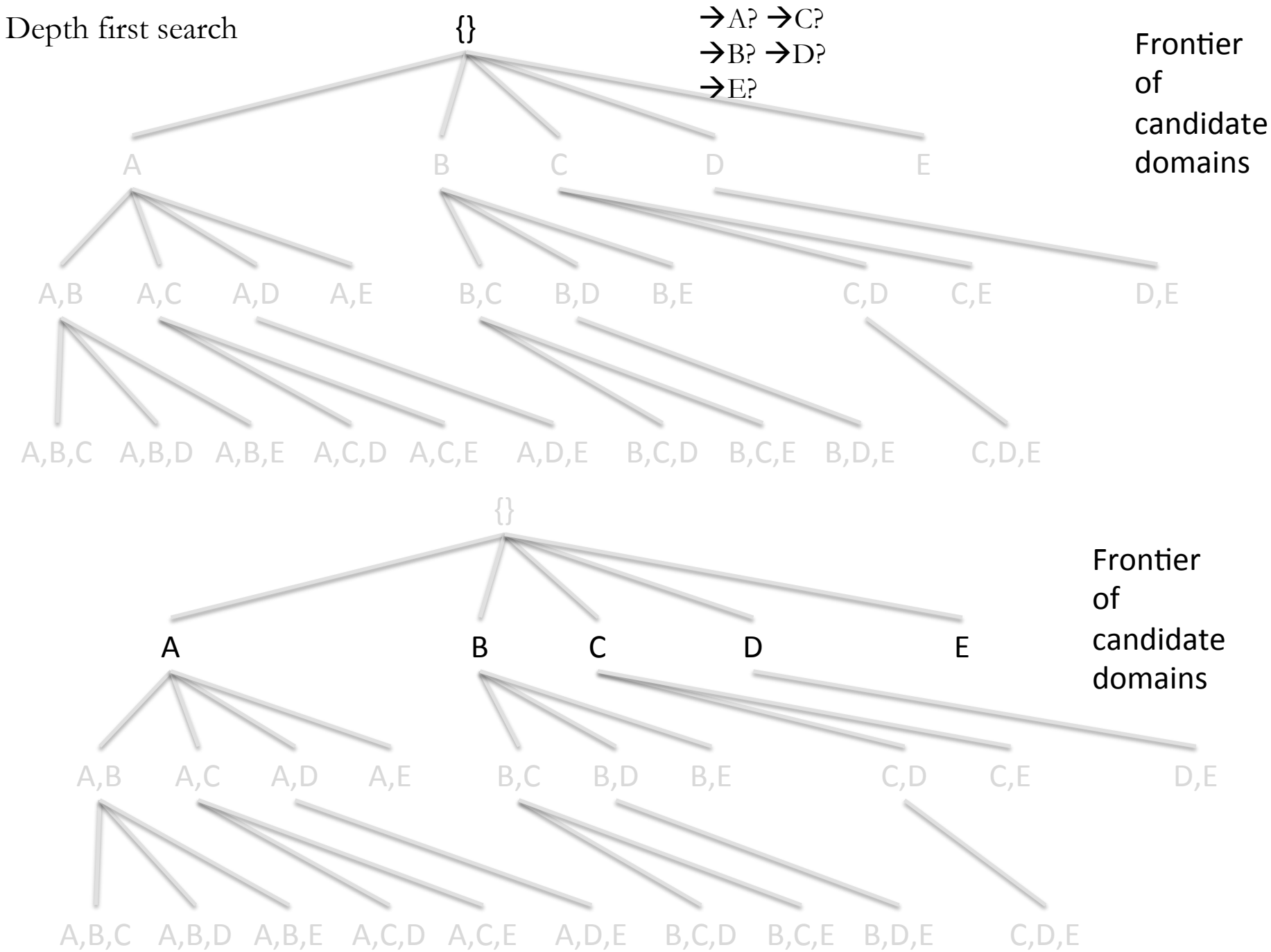


Breadth first search

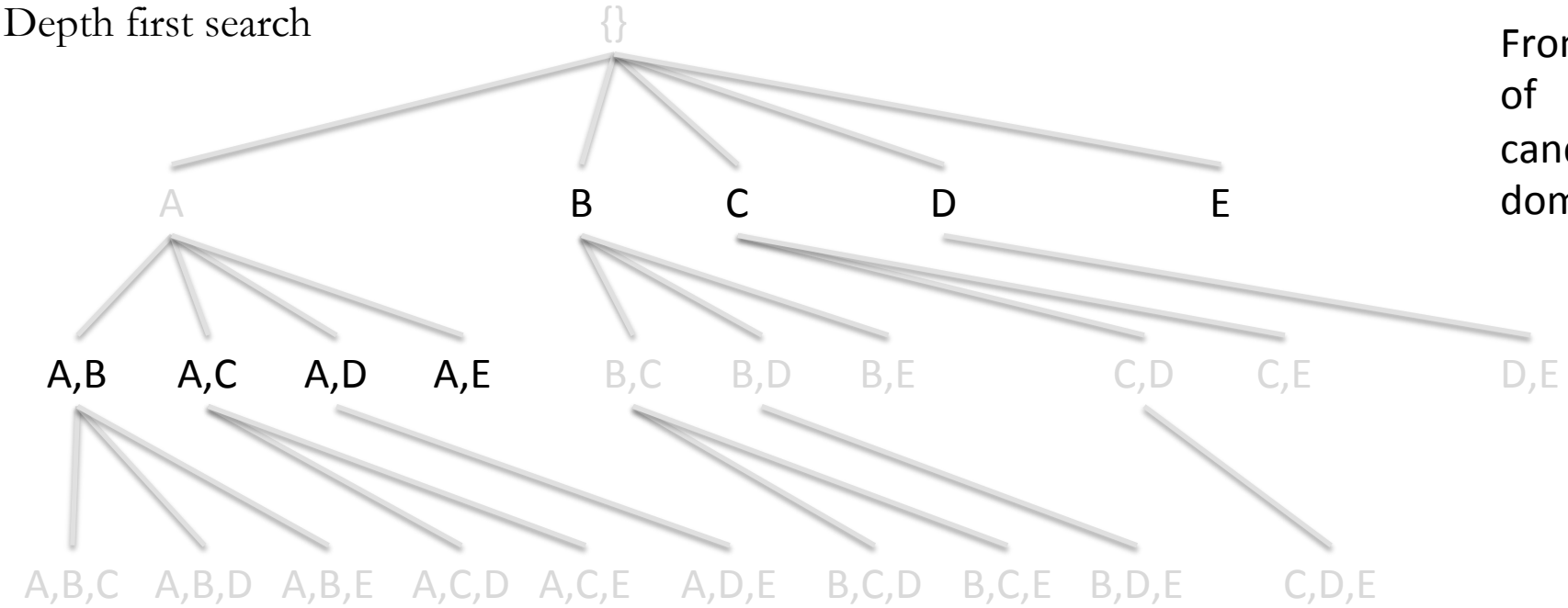


Frontier of candidate domains

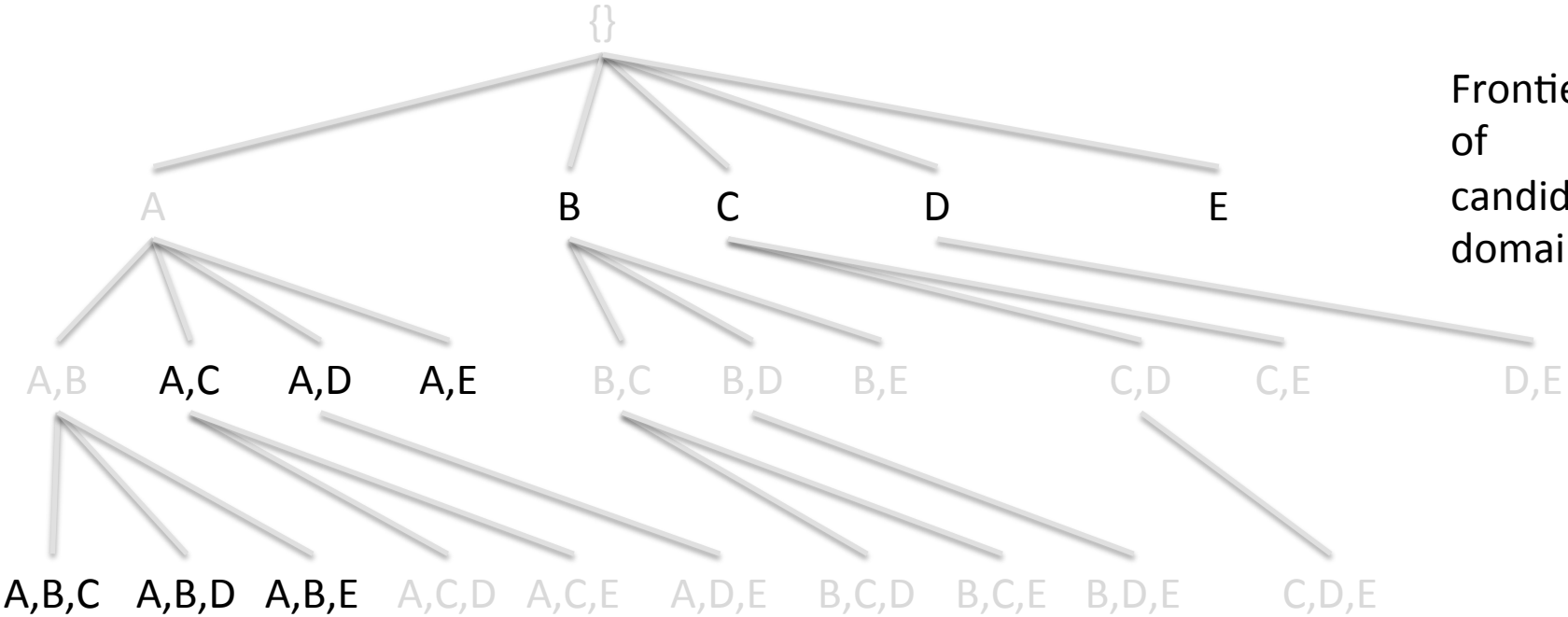



Depth first search
{}
→A? →C?
→B? →D?
→E?
Frontier of candidate domains
A
B
C
D
E
A,B
A,C
A,D
A,E
B,C
B,D
B,E
C,D
C,E
D,E
A,B,C
A,B,D
A,B,E
A,C,D
A,C,E
A,D,E
B,C,D
B,C,E
B,D,E
C,D,E
{}
Frontier of candidate domains
A
B
C
D
E
A,B
A,C
A,D
A,E
B,C
B,D
B,E
C,D
C,E
D,E
A,B,C
A,B,D
A,B,E
A,C,D
A,C,E
A,D,E
B,C,D
B,C,E
B,D,E
C,D,E

Depth first search

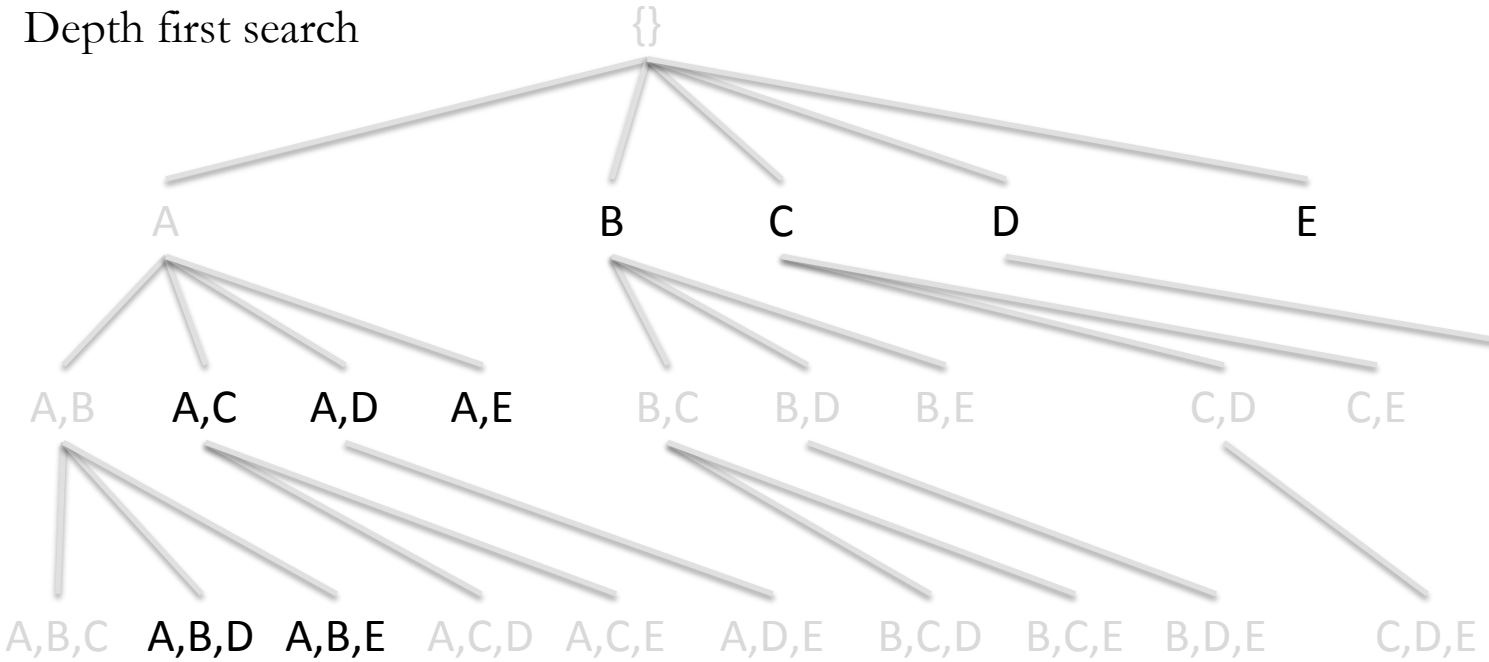


Frontier of candidate domains

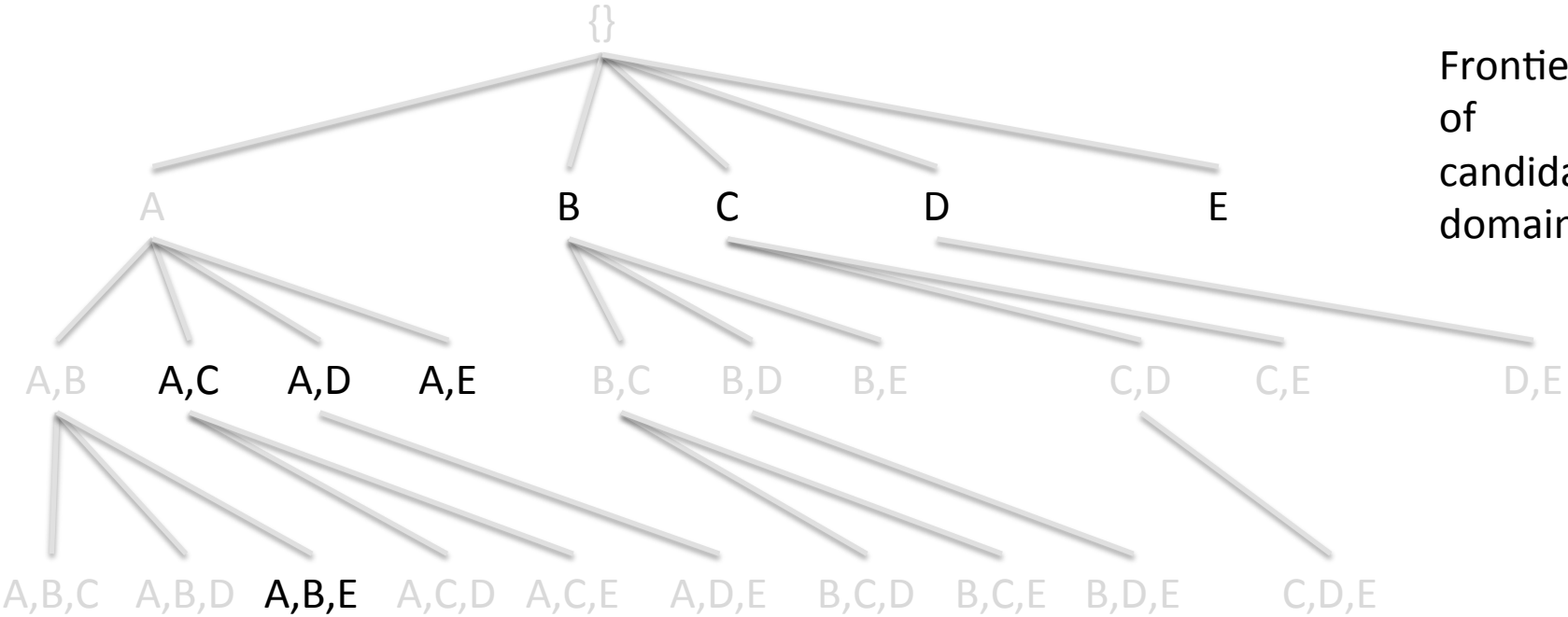


Frontier of candidate domains

Depth first search

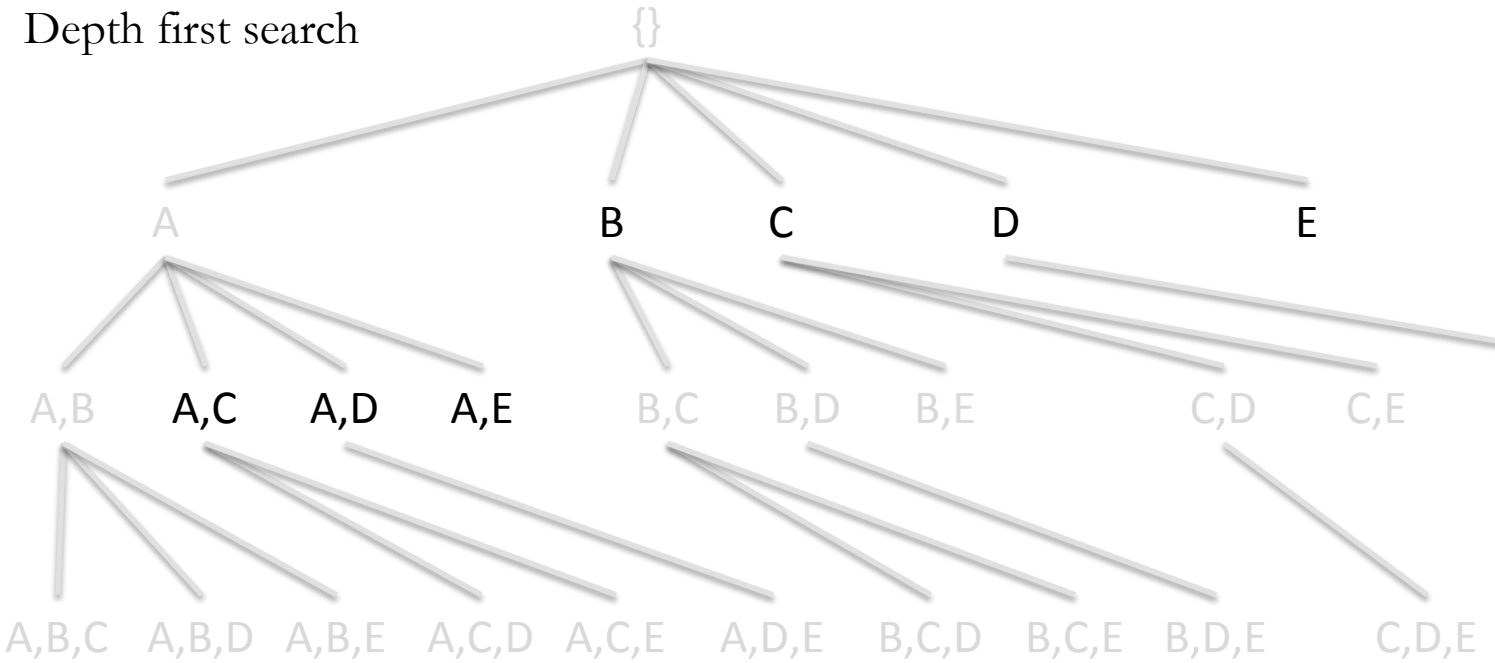


Frontier of candidate domains

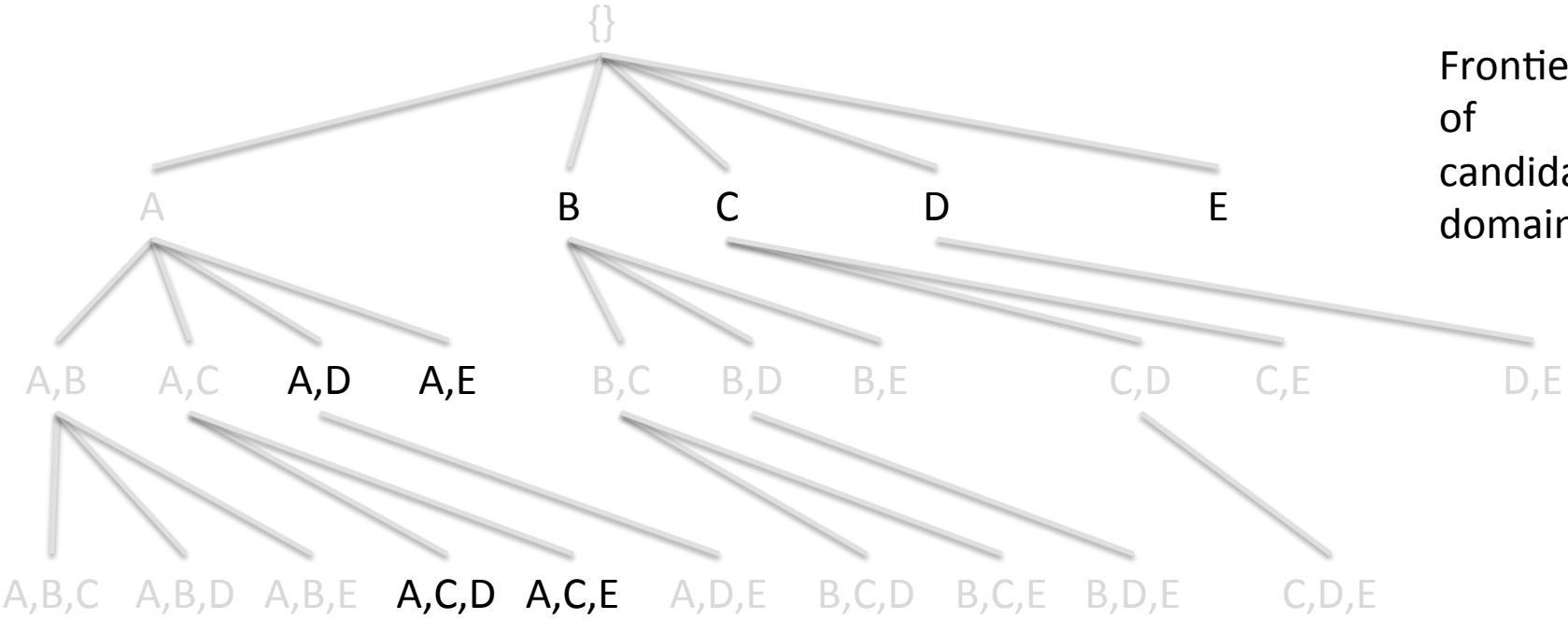


Frontier of candidate domains

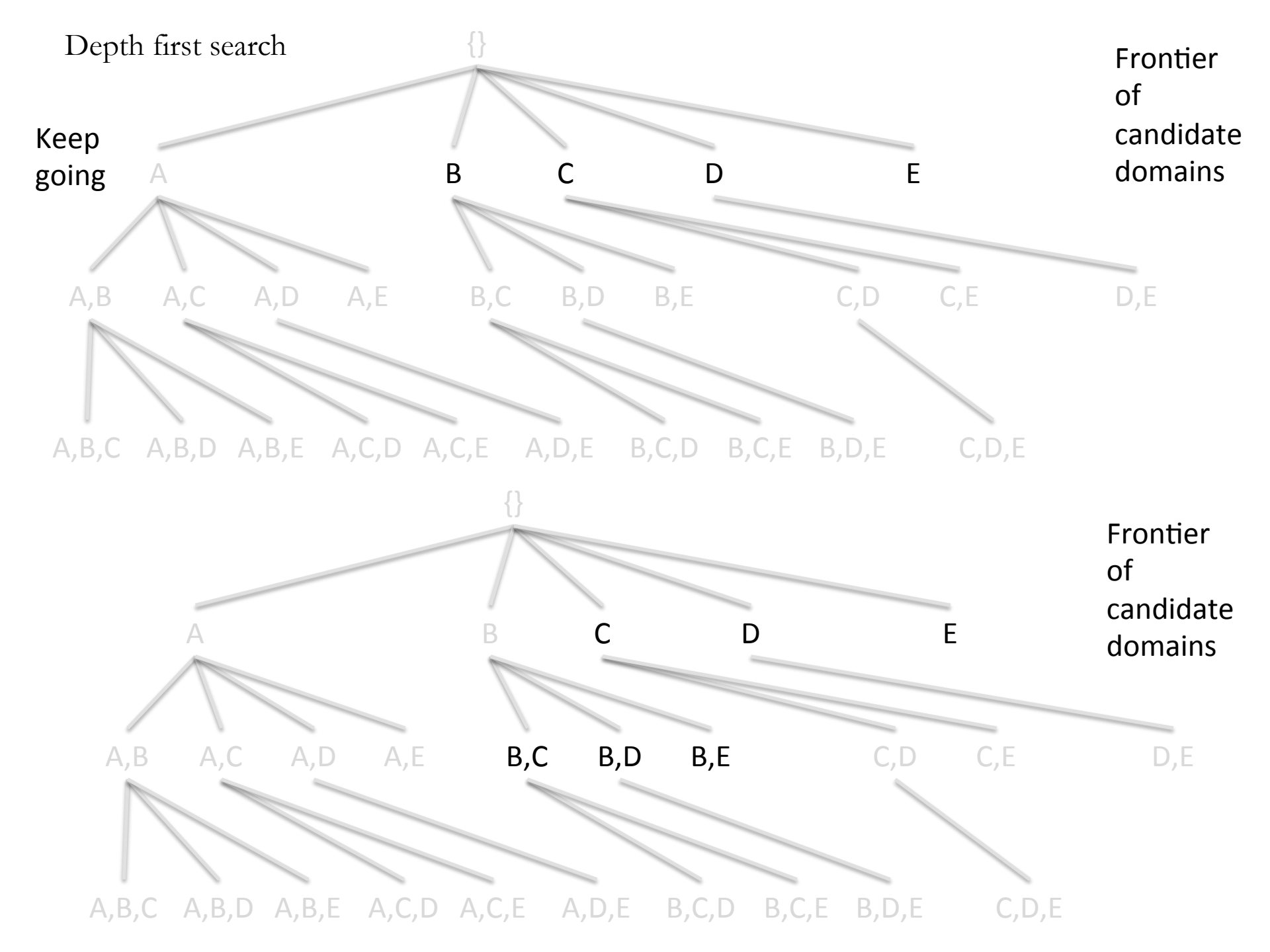
Depth first search



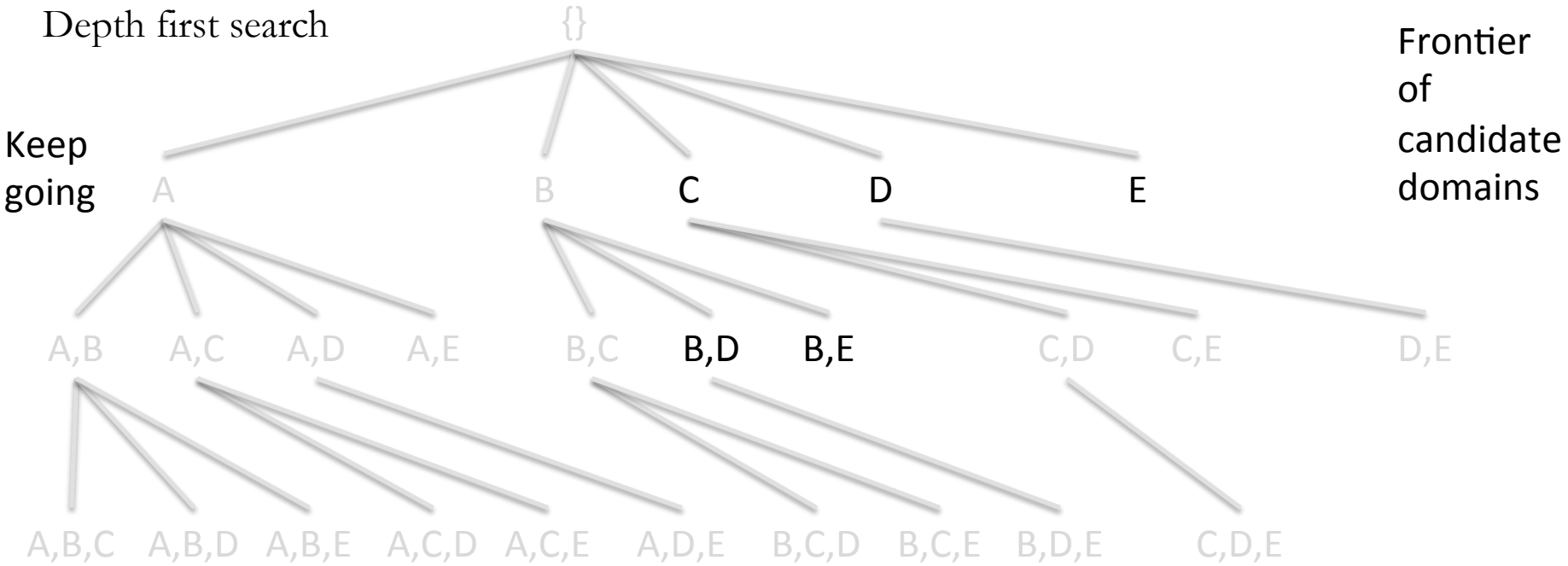
Frontier of candidate domains



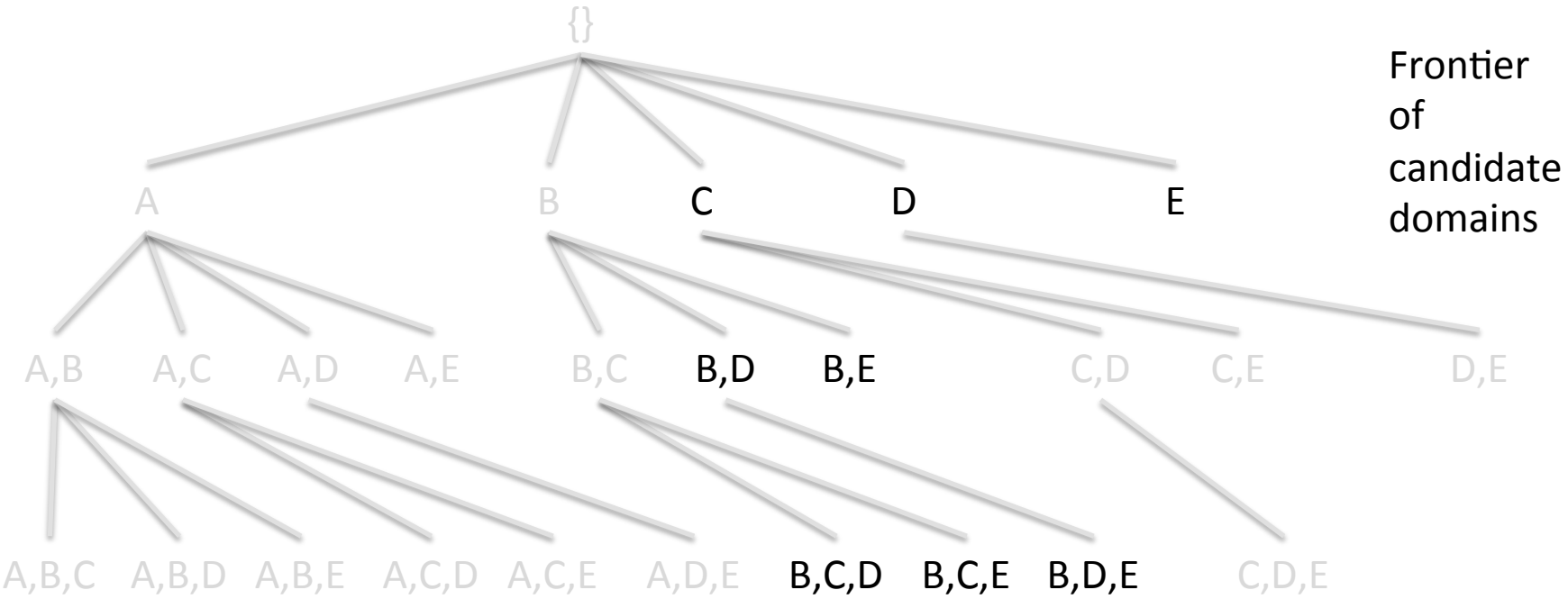
Frontier of candidate domains


Depth first search
{}
Keep going
A
B
C
D
E
Frontier of candidate domains
A,B
A,C
A,D
A,E
B,C
B,D
B,E
C,D
C,E
D,E
A,B,C
A,B,D
A,B,E
A,C,D
A,C,E
A,D,E
B,C,D
B,C,E
B,D,E
C,D,E
{}
A
B
C
D
E
Frontier of candidate domains
A,B
A,C
A,D
A,E
B,C
B,D
B,E
C,D
C,E
D,E
A,B,C
A,B,D
A,B,E
A,C,D
A,C,E
A,D,E
B,C,D
B,C,E
B,D,E
C,D,E

Depth first search

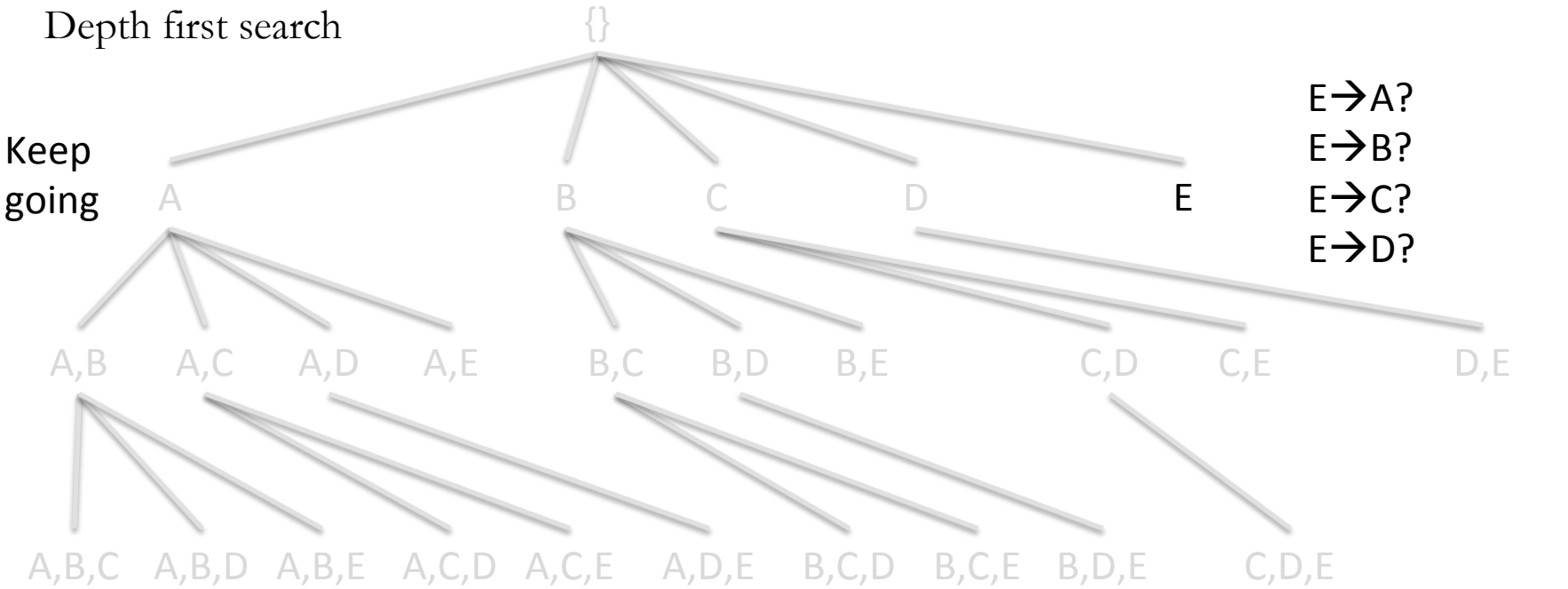


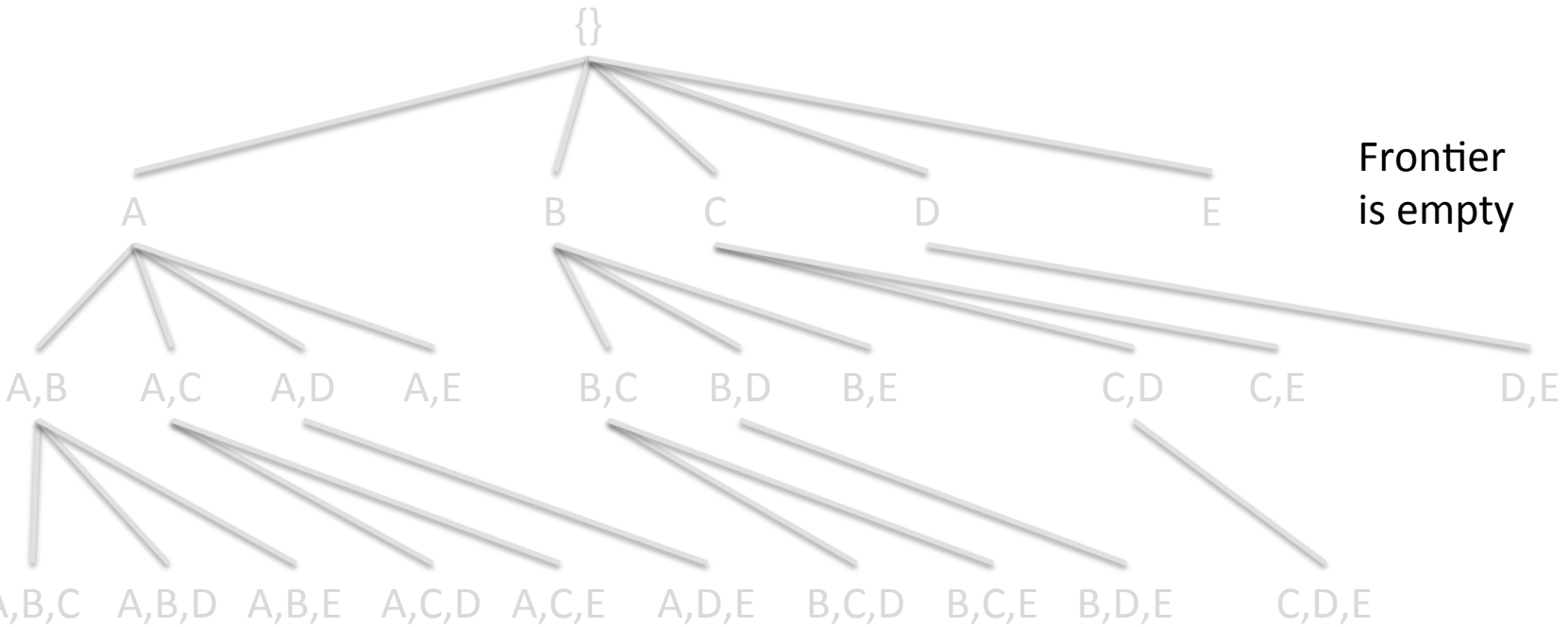
Frontier of candidate domains



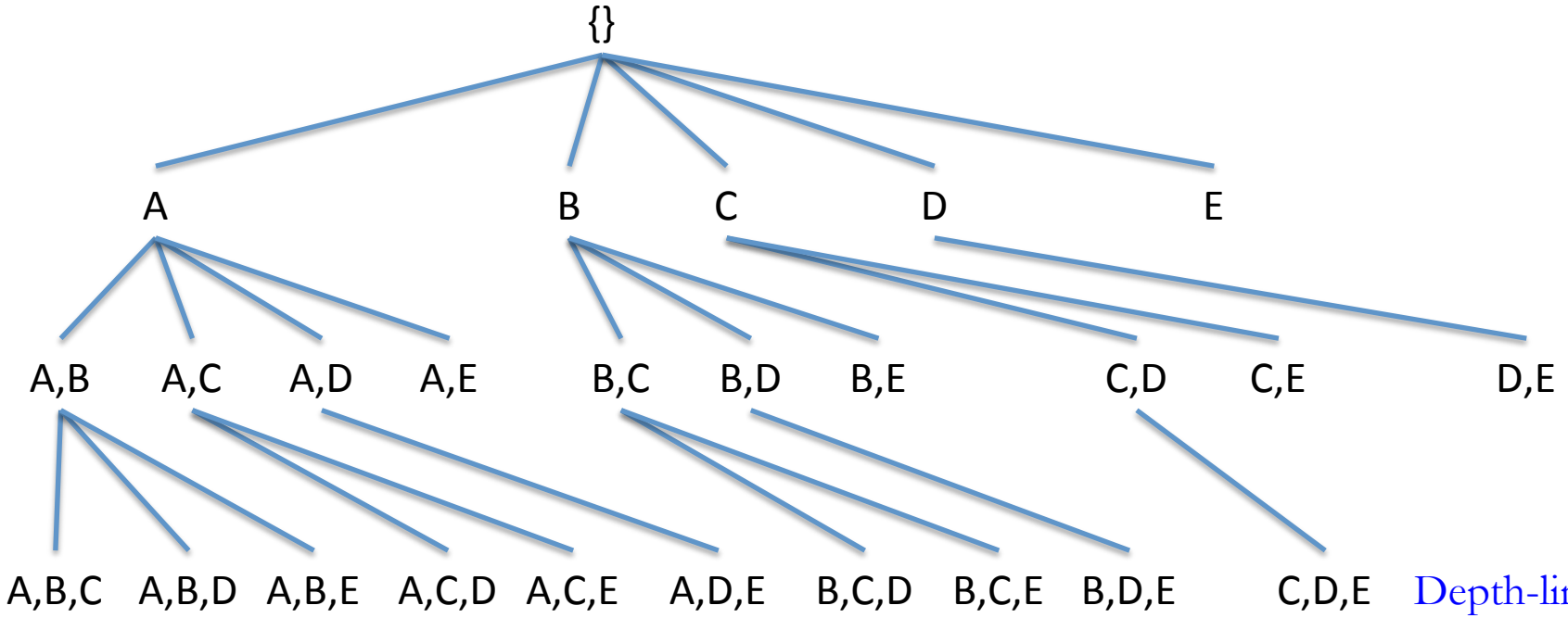
Frontier of candidate domains

Depth first search





Instead of learning only perfectly consistent FDs, beneficial to learn approximate FDs (almost perfectly consistent)This



A,B,C → D?

Number of rows adhering to dominant pattern with Range attribute (numerator)

Number of rows adhering to Domain patterns(denominator)

((a1,b3,c4:100 rows),((d1:98 rows), (d3: 2 rows))
((a2,b2,c1:43 rows),((d2:42 rows)(d1: 1 row))
((a3,b1,c1:15 rows),((d1:15 rows))

(98+42+15)/(100+43+15)
= 155/158
= 0.98 support

If parameter *minimal-support = 0.95* then accept A,B,C→D (0.98)

Pseudo code

```
Find-Approximate-Functional-Dependencies (data-set, depth-limit, minimal-support)
    approximate-FDs ← { }   /* the empty set */
    domains-frontier ← {{}} /* the set containing the empty set */
    WHILE domains-frontier ≠ {}
        next-domain ← Select from domains-frontier
        domains-frontier ← domains-frontier – next-domain
        FOR each attribute, Y, where Y ≠ any attribute in next-domain
            support ←compute support for FD (next-domain → Y) using data-set
            IF support >= minimum-support
                THEN approximate-FDs ← approximate-FDs + (next-domain, Y, support)
        IF |next-domain| < depth-limit
            THEN FOR all attributes, X, where X > all attributes in next-domain
                    domains-frontier ← domains-frontier + (next-domain + X)
    RETURN approximate-FDs
```

Pseudo code

```
Find-Approximate-Functional-Dependencies (data-set, depth-limit, minimal-support)
    approximate-FDs ← { }   /* the empty set */
    domains-frontier ← {{}} /* the set containing the empty set */
    WHILE domains-frontier ≠ {}
        next-domain ← Select from domains-frontier
        domains-frontier ← domains-frontier – next-domain
      FOR each attribute, Y, where Y ≠ any attribute in next-domain
          support ←compute support for FD (next-domain → Y) using data-set
          IF support >= minimum-support
             THEN approximate-FDs ← approximate-FDs + (next-domain, Y, support)
      IF |next-domain| < depth-limit
          THEN FOR all attributes, X, where X > all attributes in next-domain
                    domains-frontier ← domains-frontier + (next-domain + X)
    RETURN approximate-FDs
```

If domains-frontier is a **queue** then **breadth-first** search

Pseudo code

```
Find-Approximate-Functional-Dependencies (data-set, depth-limit, minimal-support)
    approximate-FDs ← { }   /* the empty set */
    domains-frontier ← {{}} /* the set containing the empty set */
    WHILE domains-frontier ≠ {}
        next-domain ← Select from domains-frontier
        domains-frontier ← domains-frontier – next-domain
      FOR each attribute, Y, where Y ≠ any attribute in next-domain
          support ←compute support for FD (next-domain → Y) using data-set
          IF support >= minimum-support
             THEN approximate-FDs ← approximate-FDs + (next-domain, Y, support)
      IF |next-domain| < depth-limit
          THEN FOR all attributes, X, where X > all attributes in next-domain
                    domains-frontier ← domains-frontier + (next-domain + X)
    RETURN approximate-FDs
```

If domains-frontier is a **stack** then **depth-first** search