

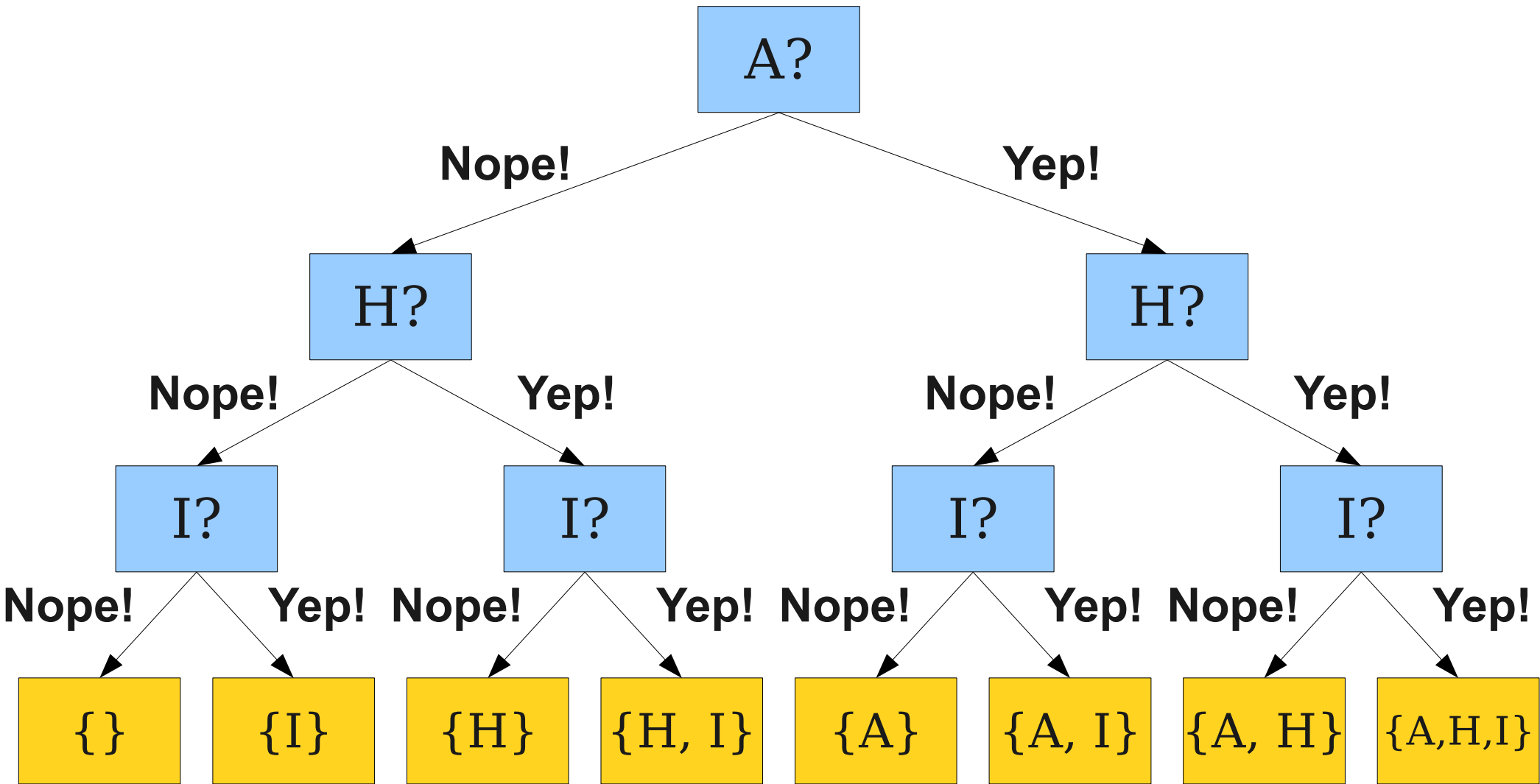
Thinking Recursively

Part IV

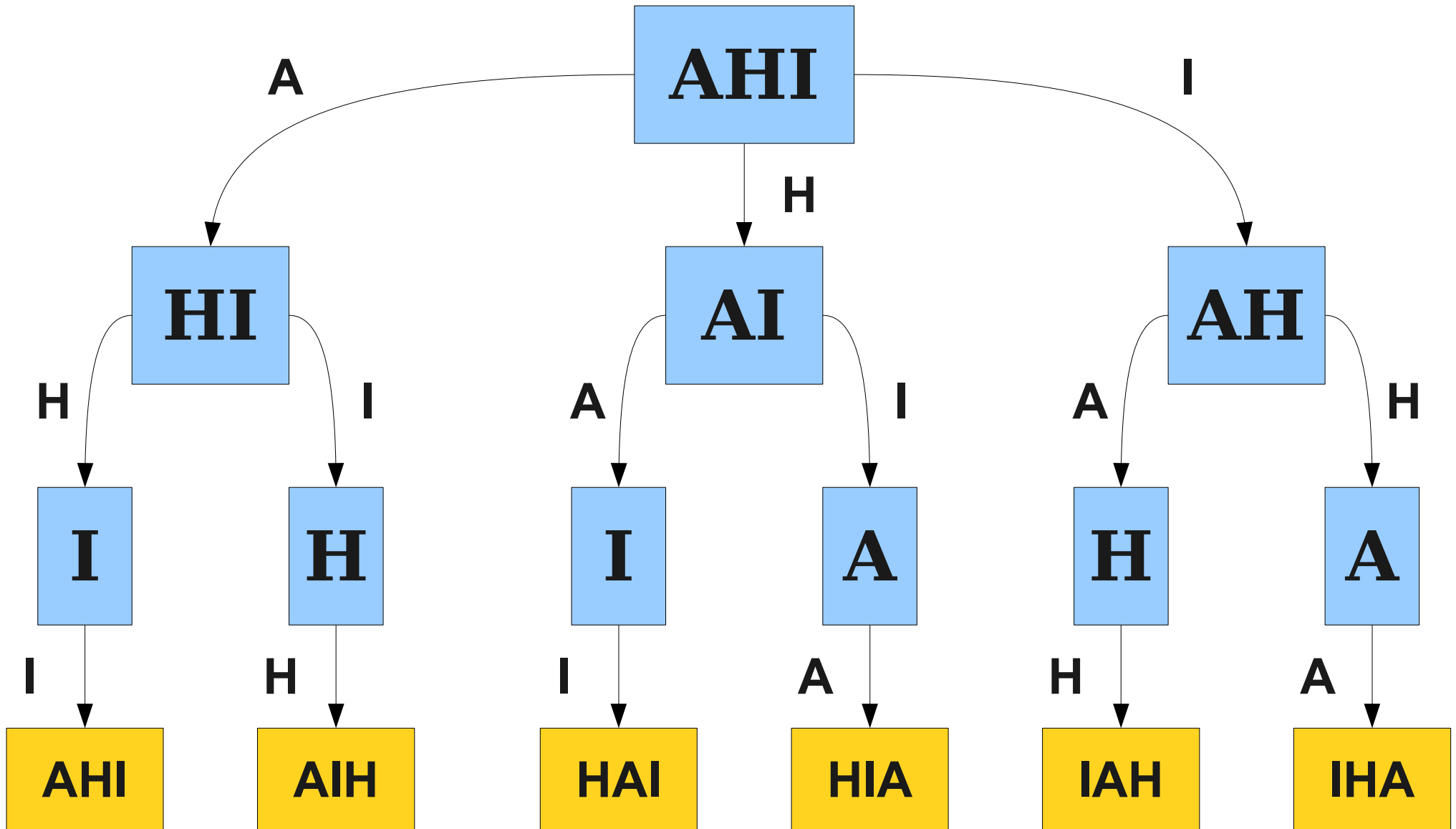
Announcements

- **Assignment 3 (Recursion!)** out now, due **Friday, May 3** at 2:15PM.
 - Assignment review hours: Tomorrow from 5:30PM – 6:30PM in Gates B12.
- Stanford Women in Computer Science is holding a hackathon for people interested in learning web programming.
 - This **Sunday** from **10AM - 10PM** in the Huang Engineering Center basement.
 - Visit **<http://www.hackoverflow.org>** for more details.
 - Everyone is welcome!
- Keith's office hours tomorrow shifted to 2:30PM – 4:30PM in Gates 178.

A Decision Tree



A Decision Tree



Generating Combinations

- Suppose that we want to find every way to choose exactly **one** element from a set.
- We could do something like this:

```
foreach (int x in mySet) {  
    cout << x << endl;  
}
```

Generating Combinations

- Suppose that we want to find every way to choose exactly **two** elements from a set.
- We could do something like this:

```
foreach (int x in mySet) {  
    foreach (int y in mySet) {  
        if (x != y) {  
            cout << x << ", " << y << endl;  
        }  
    }  
}
```

Generating Combinations

- Suppose that we want to find every way to choose exactly **three** elements from a set.
- We could do something like this:

```
foreach (int x in mySet) {  
    foreach (int y in mySet) {  
        foreach (int z in mySet) {  
            if (x != y && x != z && y != z) {  
                cout << x << ", " << y << ", " << z << endl;  
            }  
        }  
    }  
}
```

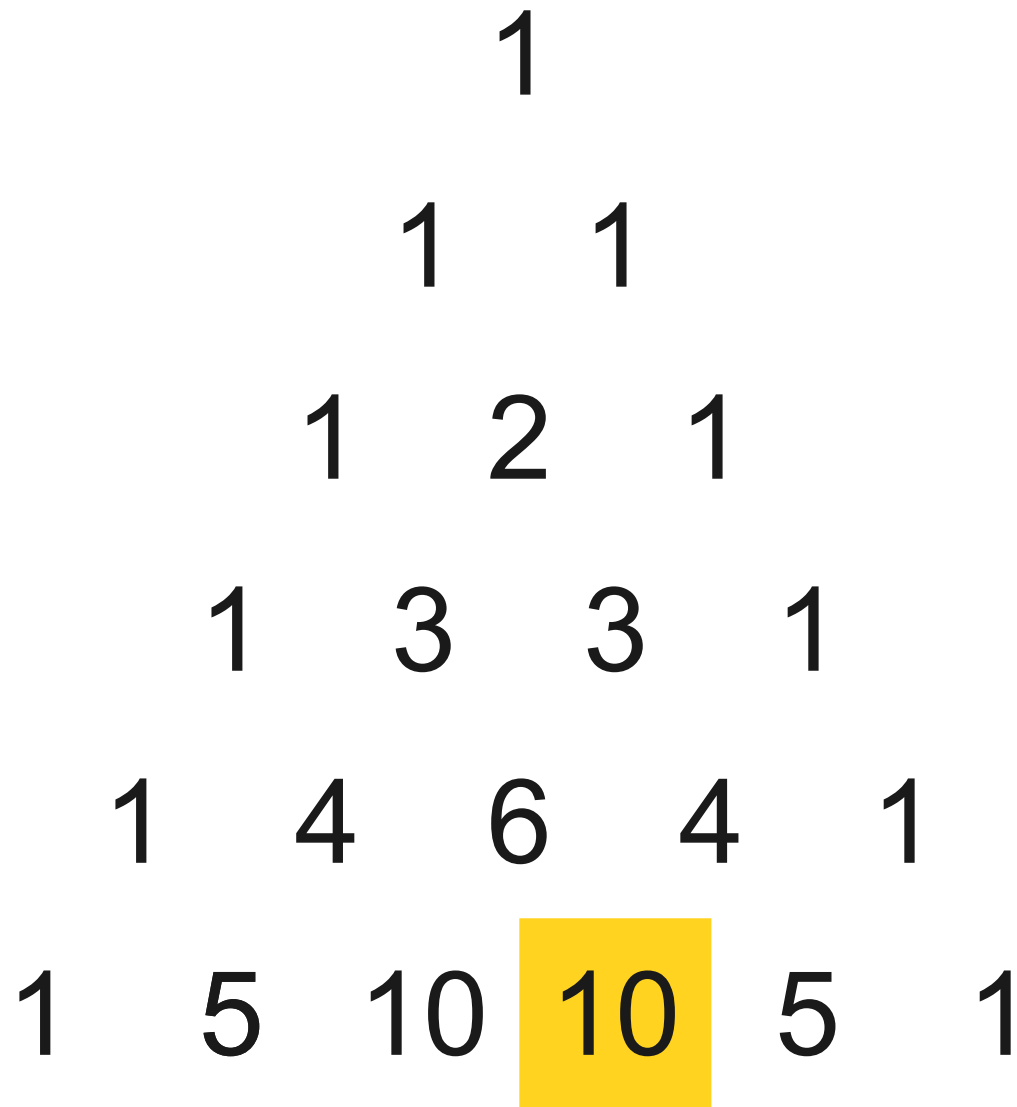
Generating Combinations

- If we know how many elements we want in advance, we can always just nest a whole bunch of loops.
- But what if we don't know in advance?

Pascal's Triangle Revisited

1					
1		1			
1		2	1		
1		3	3	1	
1	4	6	4	1	
1	5	10	10	5	1

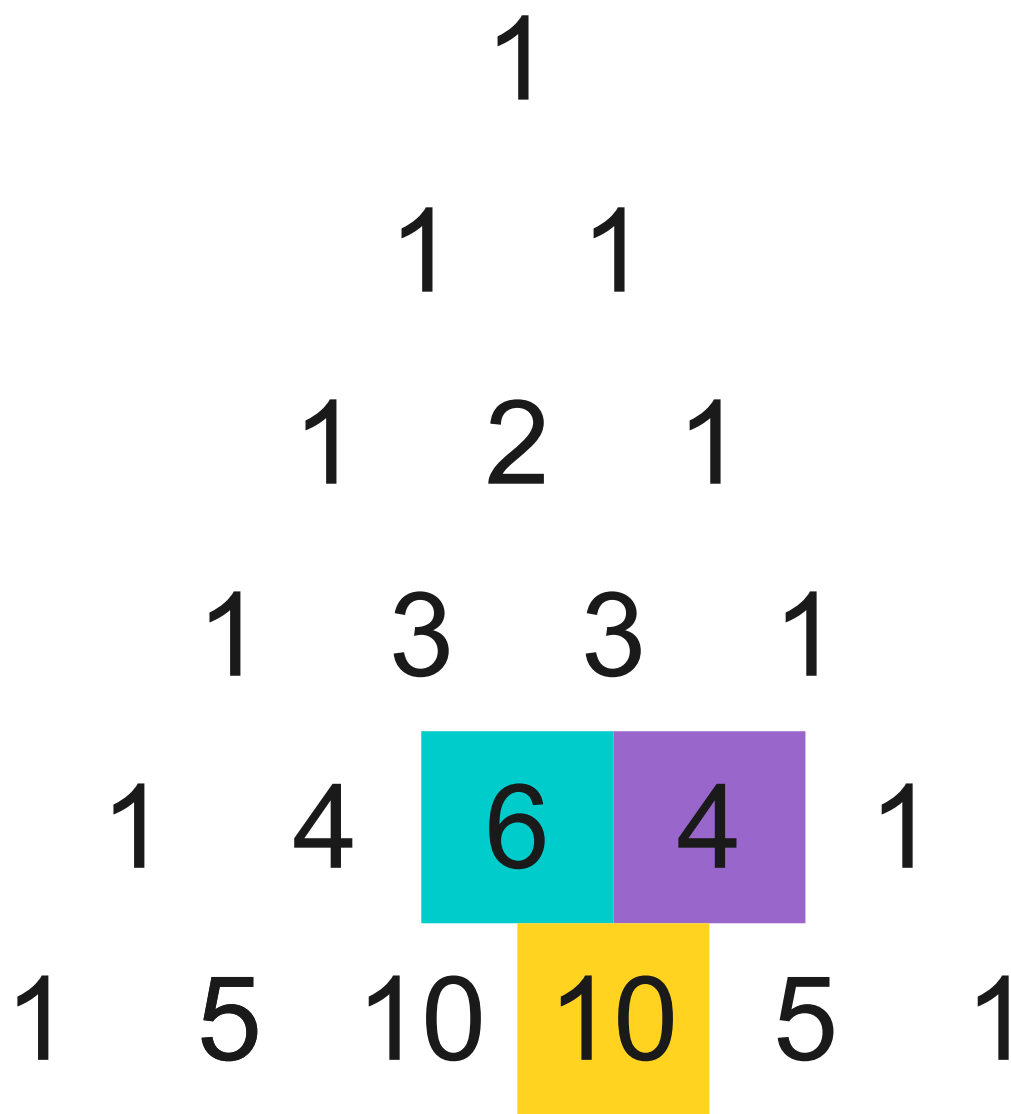
Pascal's Triangle Revisited



A Pascal's Triangle diagram with 6 rows. The numbers are arranged in a triangular shape. The second '10' in the bottom row is highlighted with a yellow background.

			1			
		1		1		
	1		2		1	
	1	3		3		1
	1	4	6	4		1
1	5	10	10	5		1

Pascal's Triangle Revisited



Pascal's Triangle Revisited

(0, 0)

(0, 1) (1, 1)

(0, 2) (1, 2) (2, 2)

(0, 3) (1, 3) (2, 3) (3, 3)

(0, 4) (1, 4) (2, 4) (3, 4) (4, 4)

(0, 5) (1, 5) (2, 5) (3, 5) (4, 5) (5, 5)



Pascal's Triangle Revisited

(0, 0)

(0, 1) (1, 1)

(0, 2) (1, 2) (2, 2)

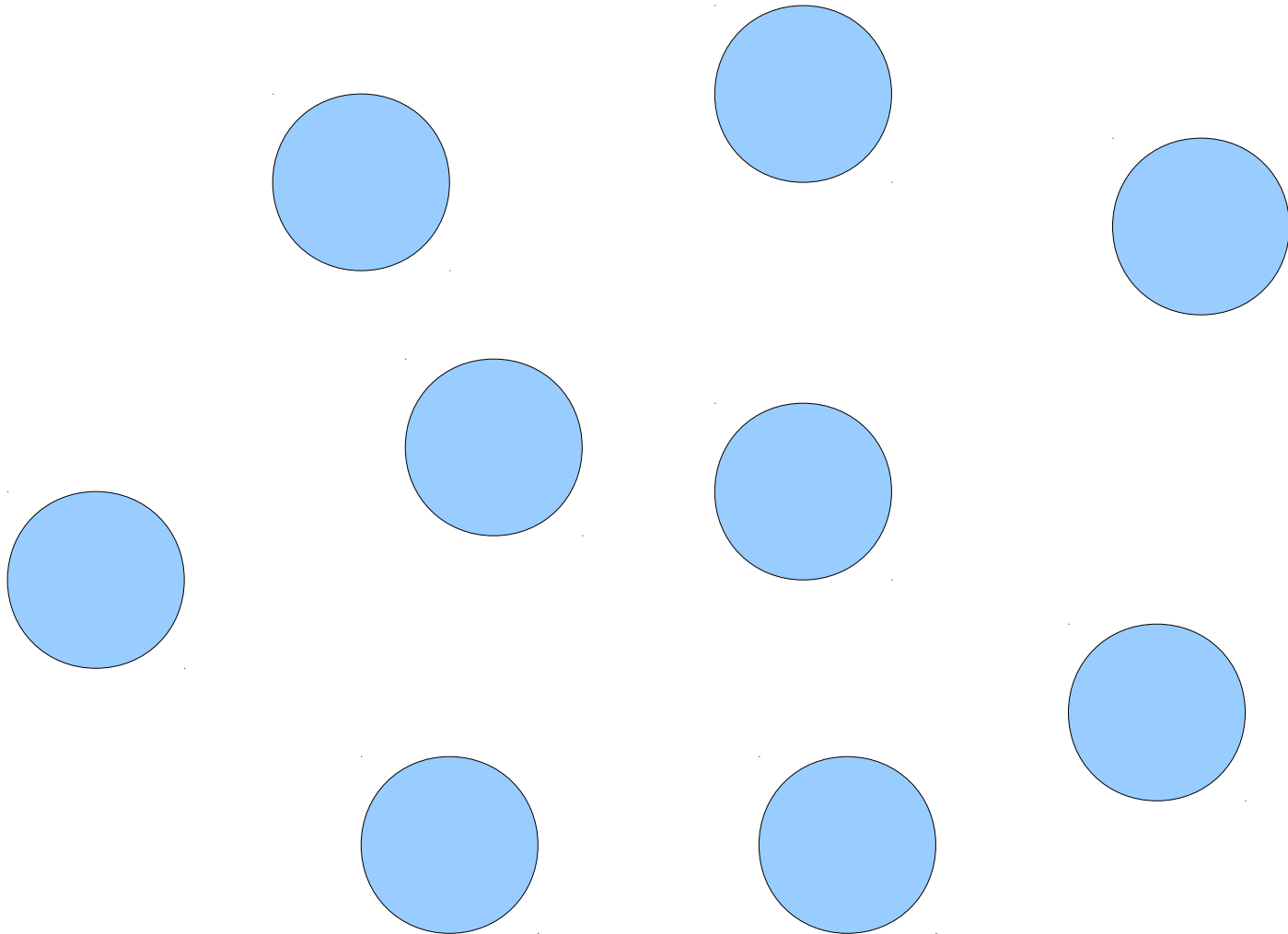
(0, 3) (1, 3) (2, 3) (3, 3)

(0, 4) (1, 4) (2, 4) (3, 4) (4, 4)

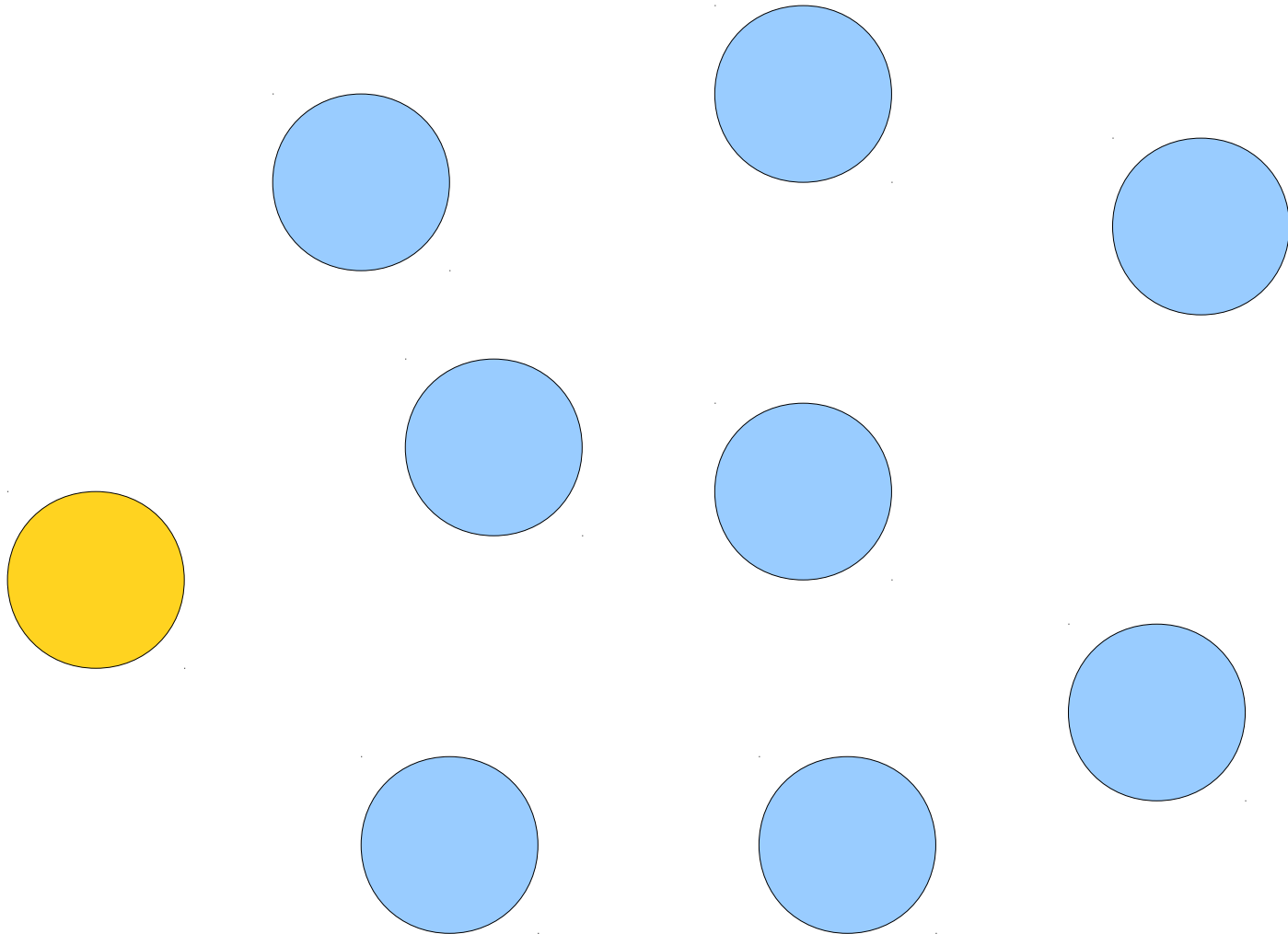
(0, 5) (1, 5) (2, 5) (3, 5) (4, 5) (5, 5)

What's up
with that?

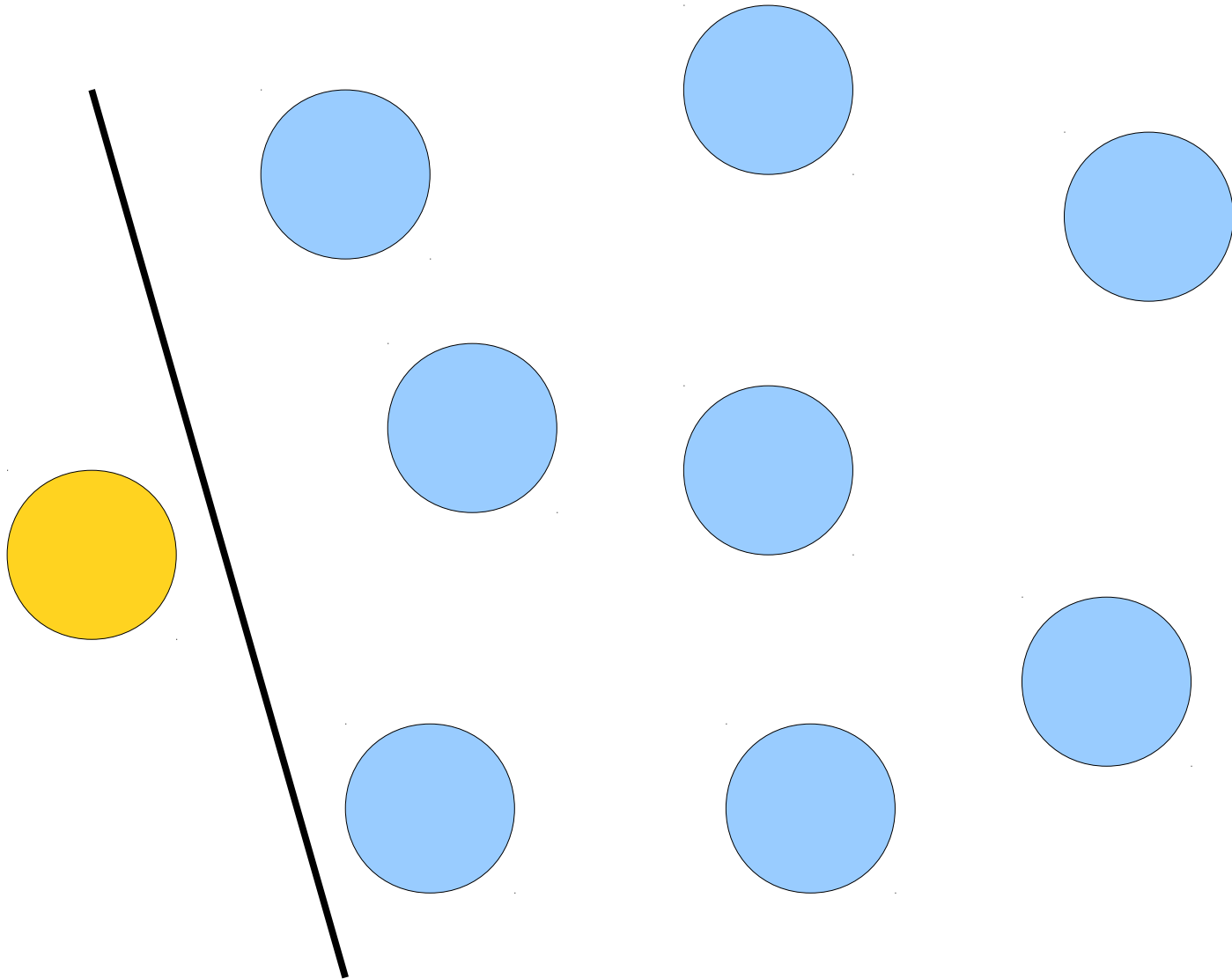
Generating Combinations



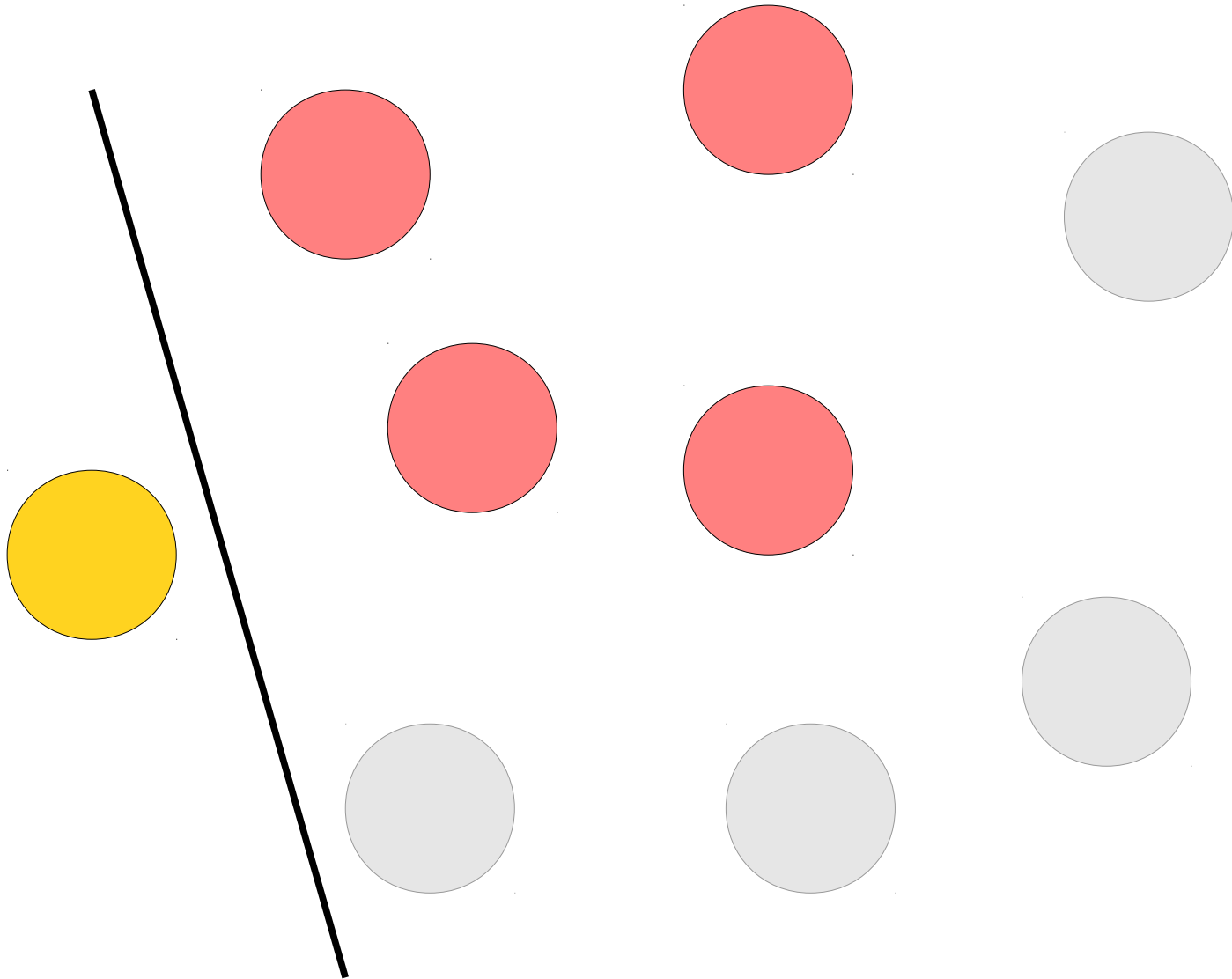
Generating Combinations



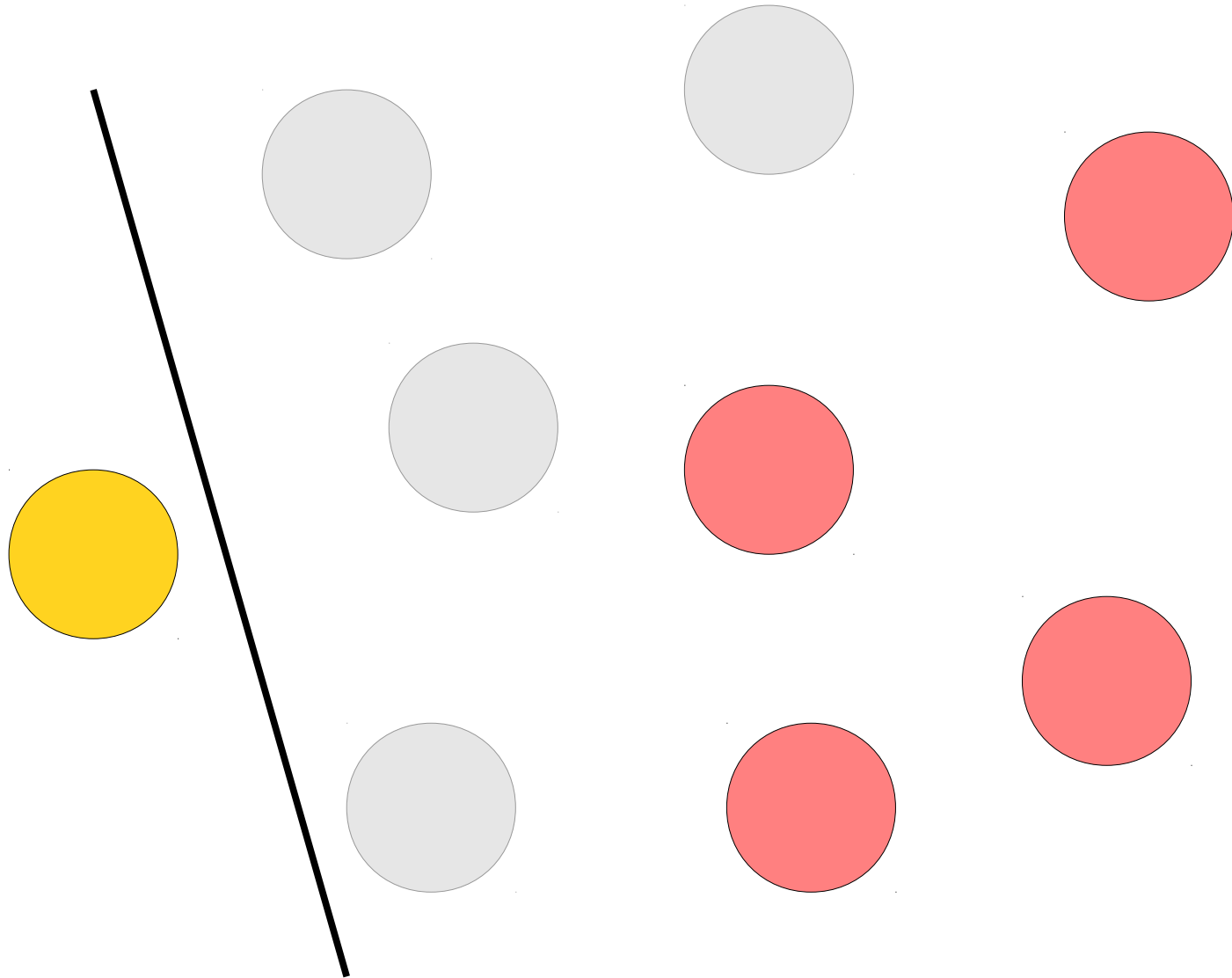
Generating Combinations



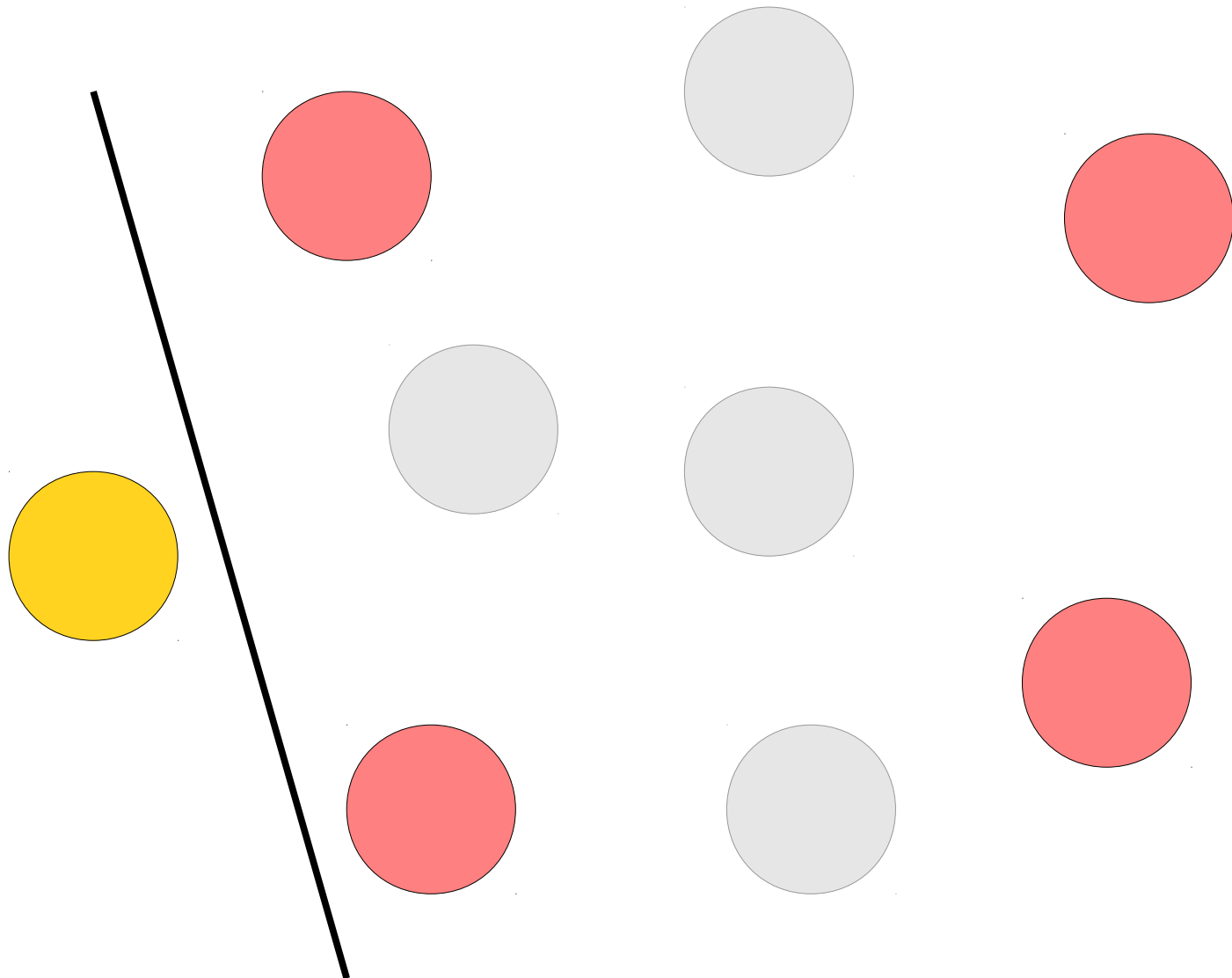
Generating Combinations



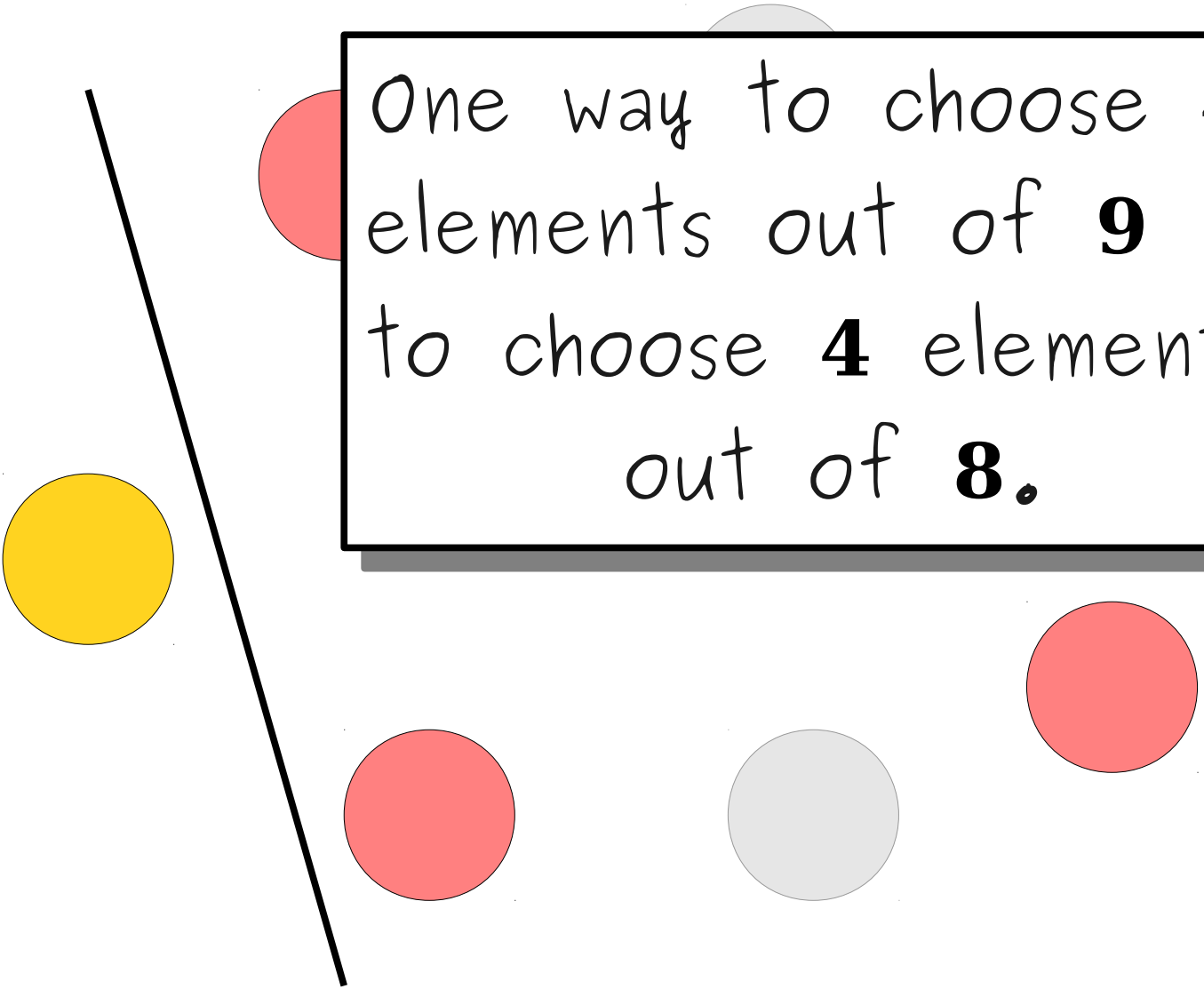
Generating Combinations



Generating Combinations

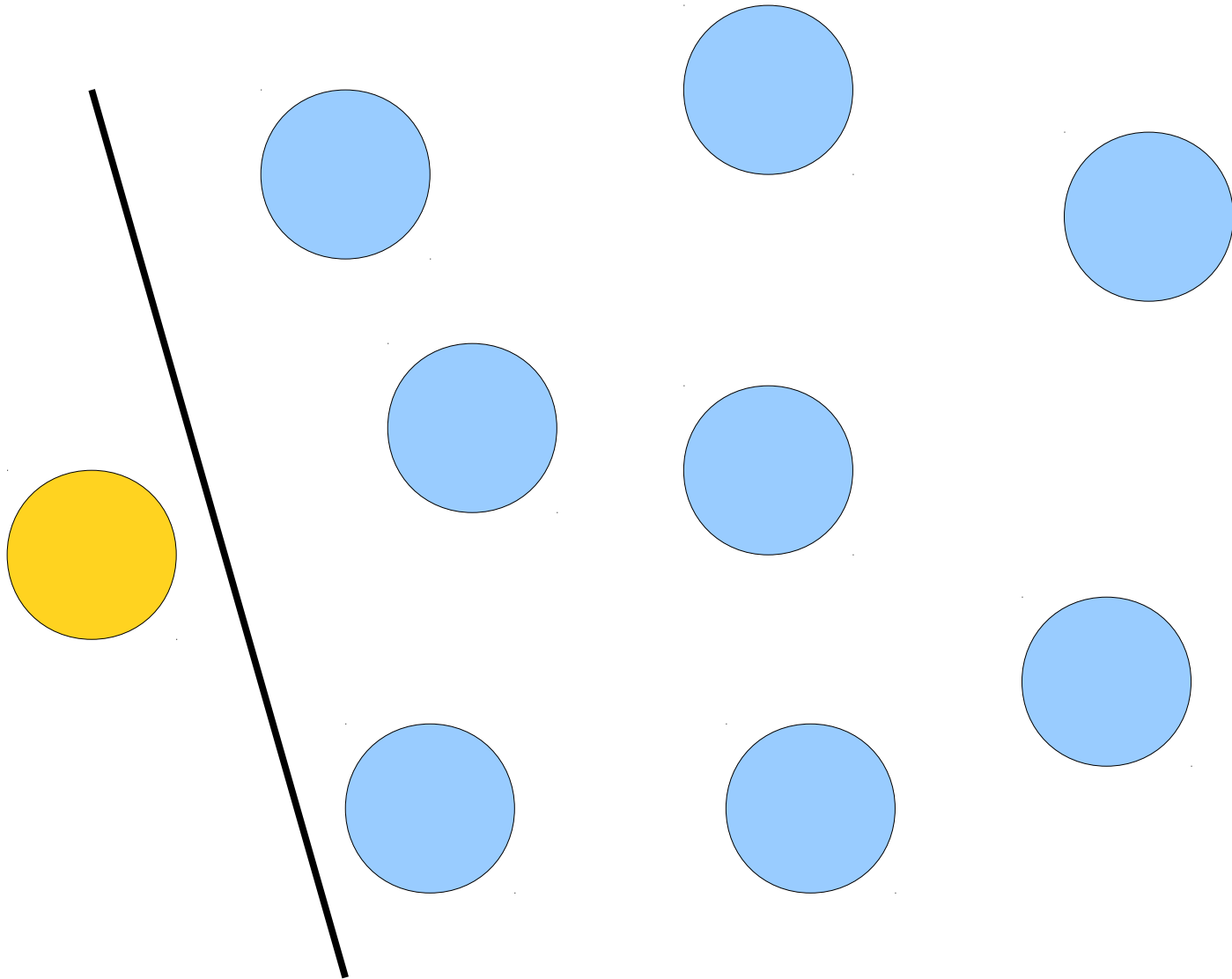


Generating Combinations

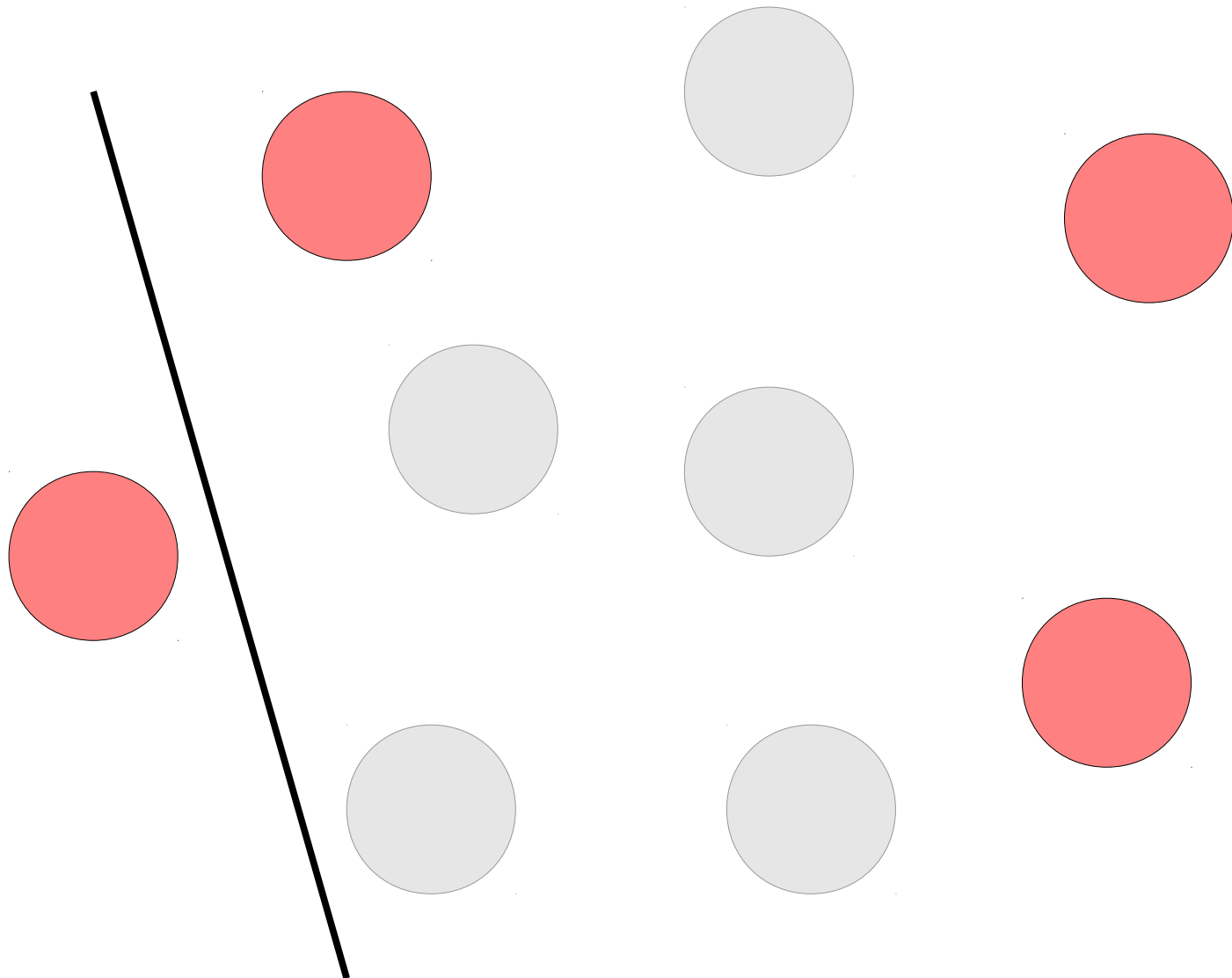


One way to choose **4** elements out of **9** is to choose **4** elements out of **8**.

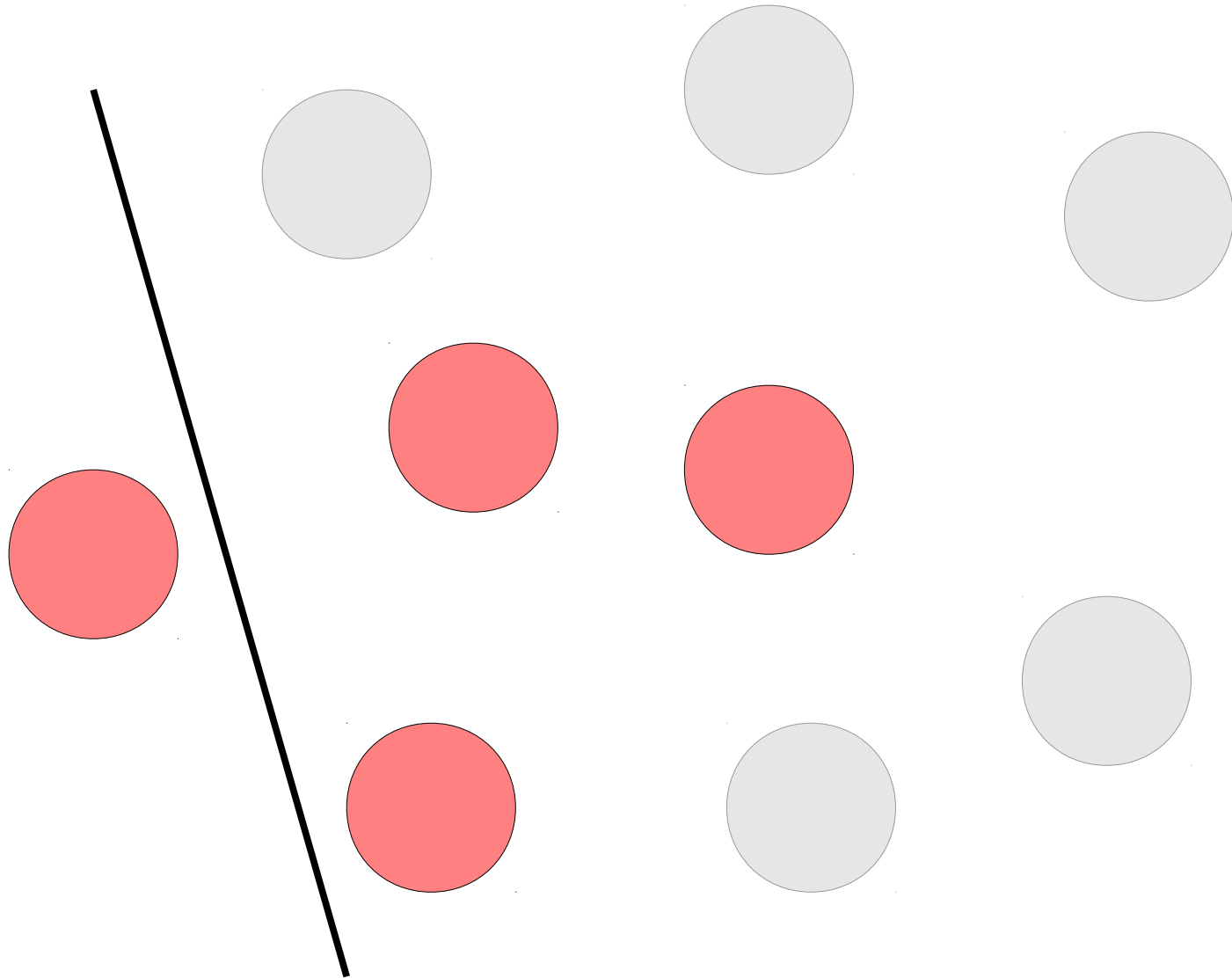
Generating Combinations



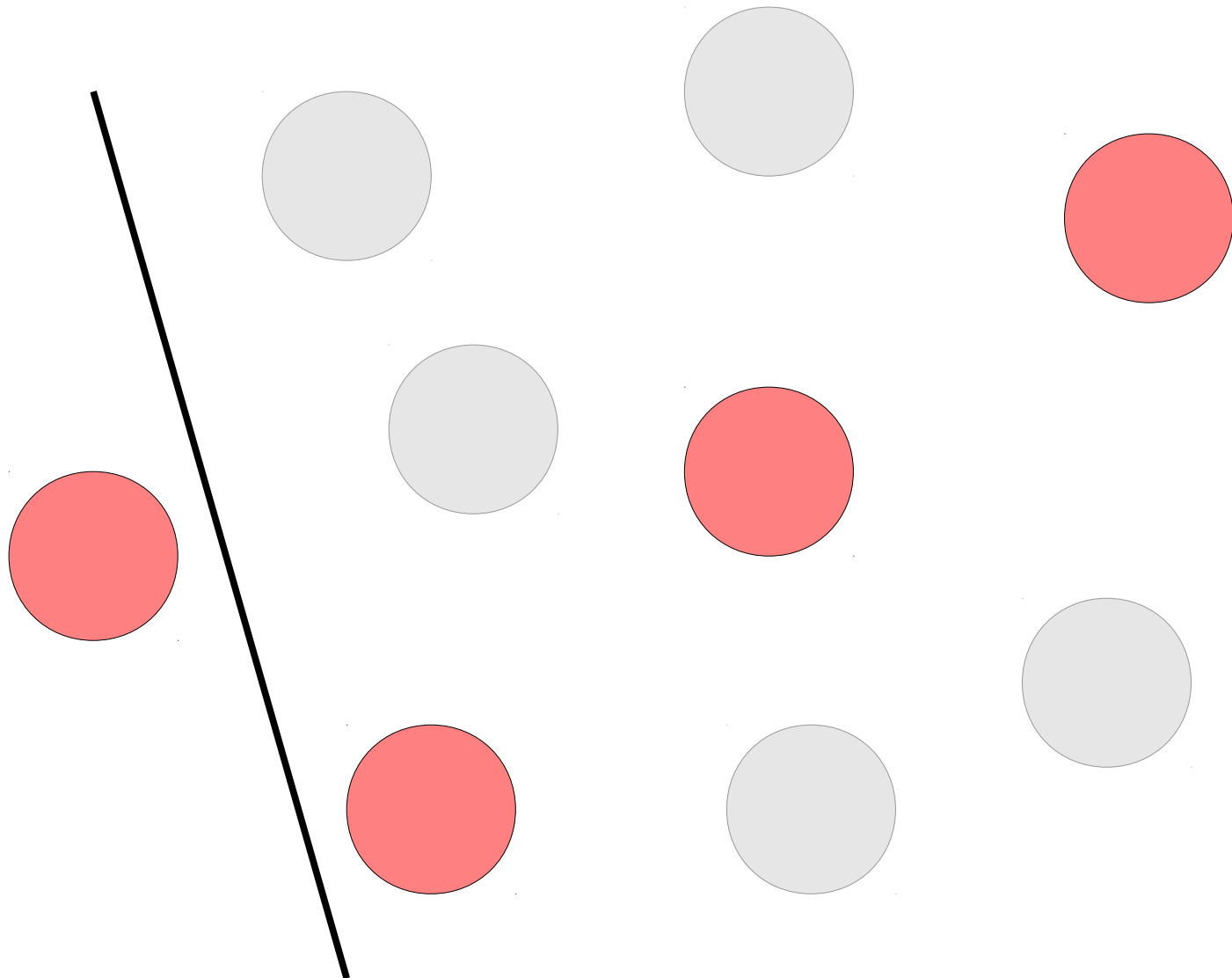
Generating Combinations



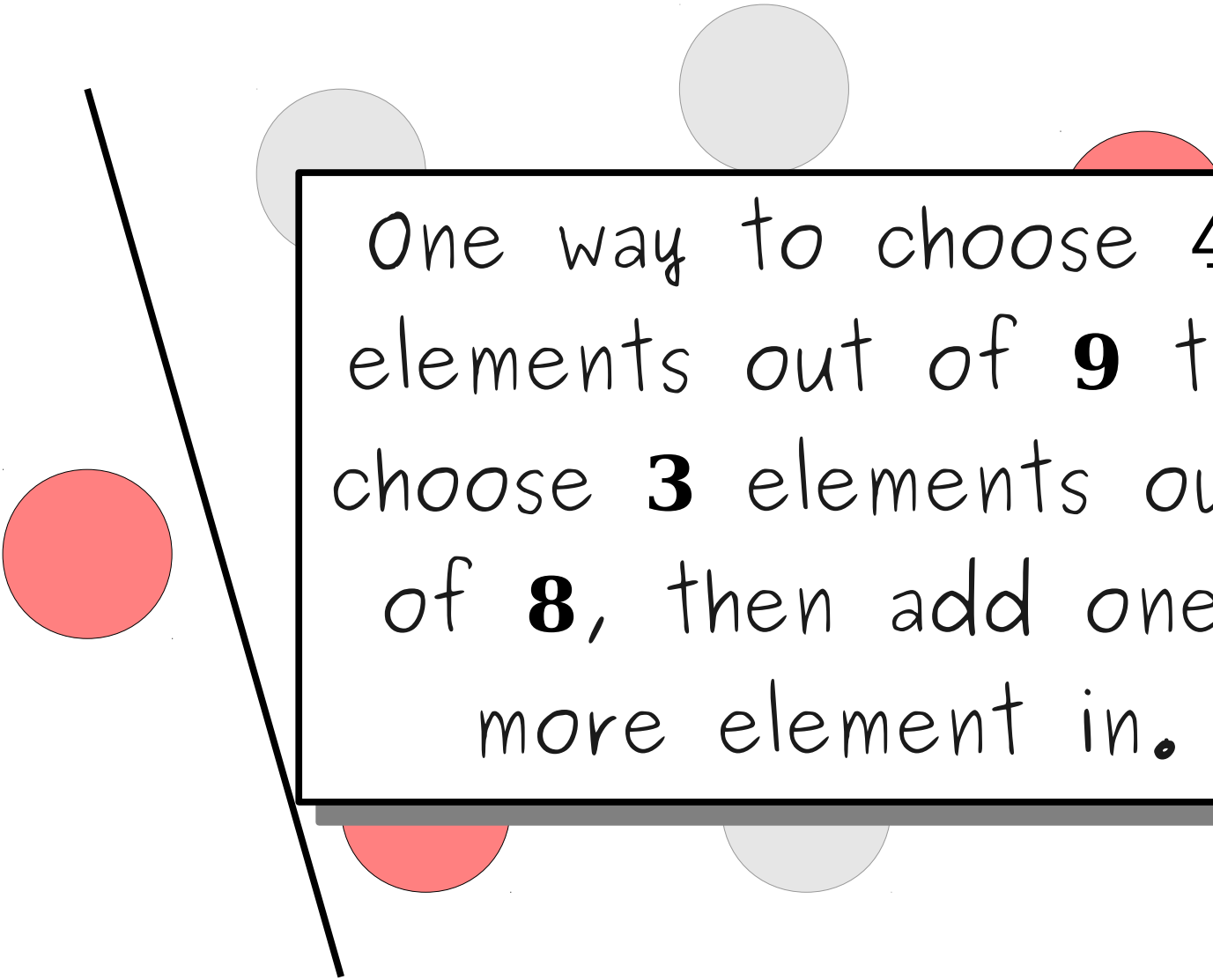
Generating Combinations



Generating Combinations



Generating Combinations

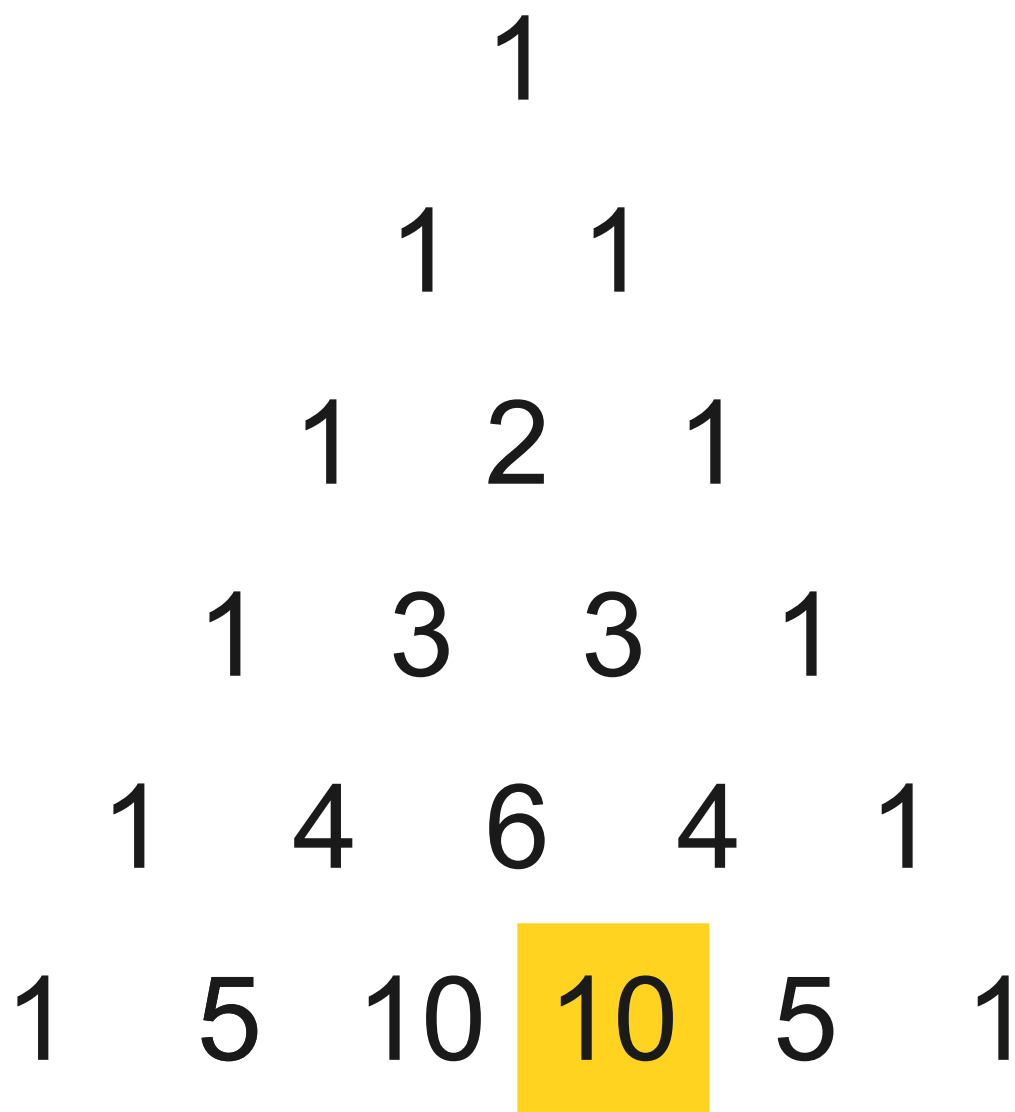


One way to choose **4** elements out of **9** to choose **3** elements out of **8**, then add one more element in.

Pascal's Triangle Revisited

1					
1		1			
1		2	1		
1		3	3	1	
1		4	6	4	1
1	5	10	10	5	1

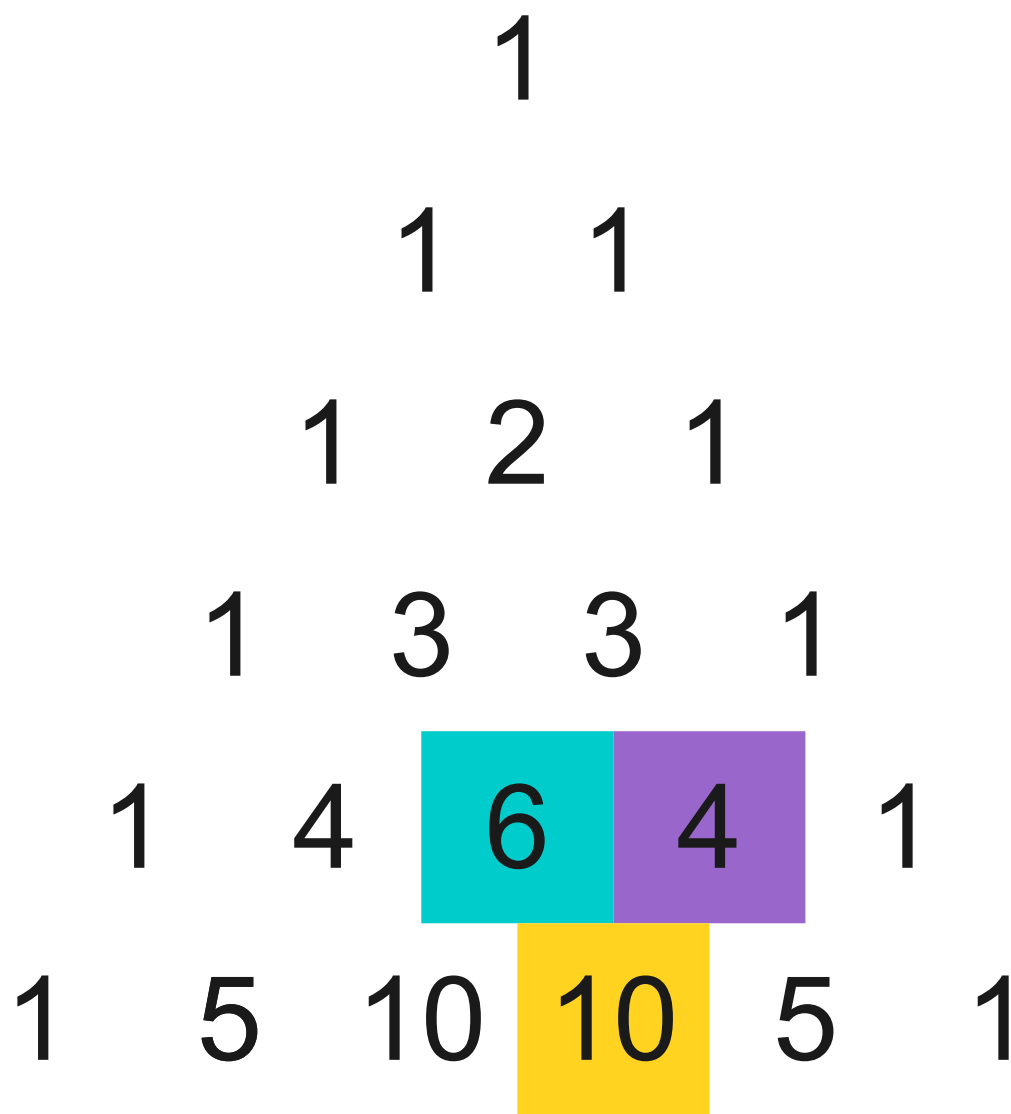
Pascal's Triangle Revisited



A Pascal's Triangle with 6 rows. The numbers are arranged in a triangular shape. The second '10' in the bottom row is highlighted with a yellow background.

			1			
		1		1		
	1		2		1	
	1	3		3		1
	1	4	6	4		1
1	5	10	10	5		1

Pascal's Triangle Revisited



Pascal's Triangle Revisited

(0, 0)

(0, 1) (1, 1)

(0, 2) (1, 2) (2, 2)

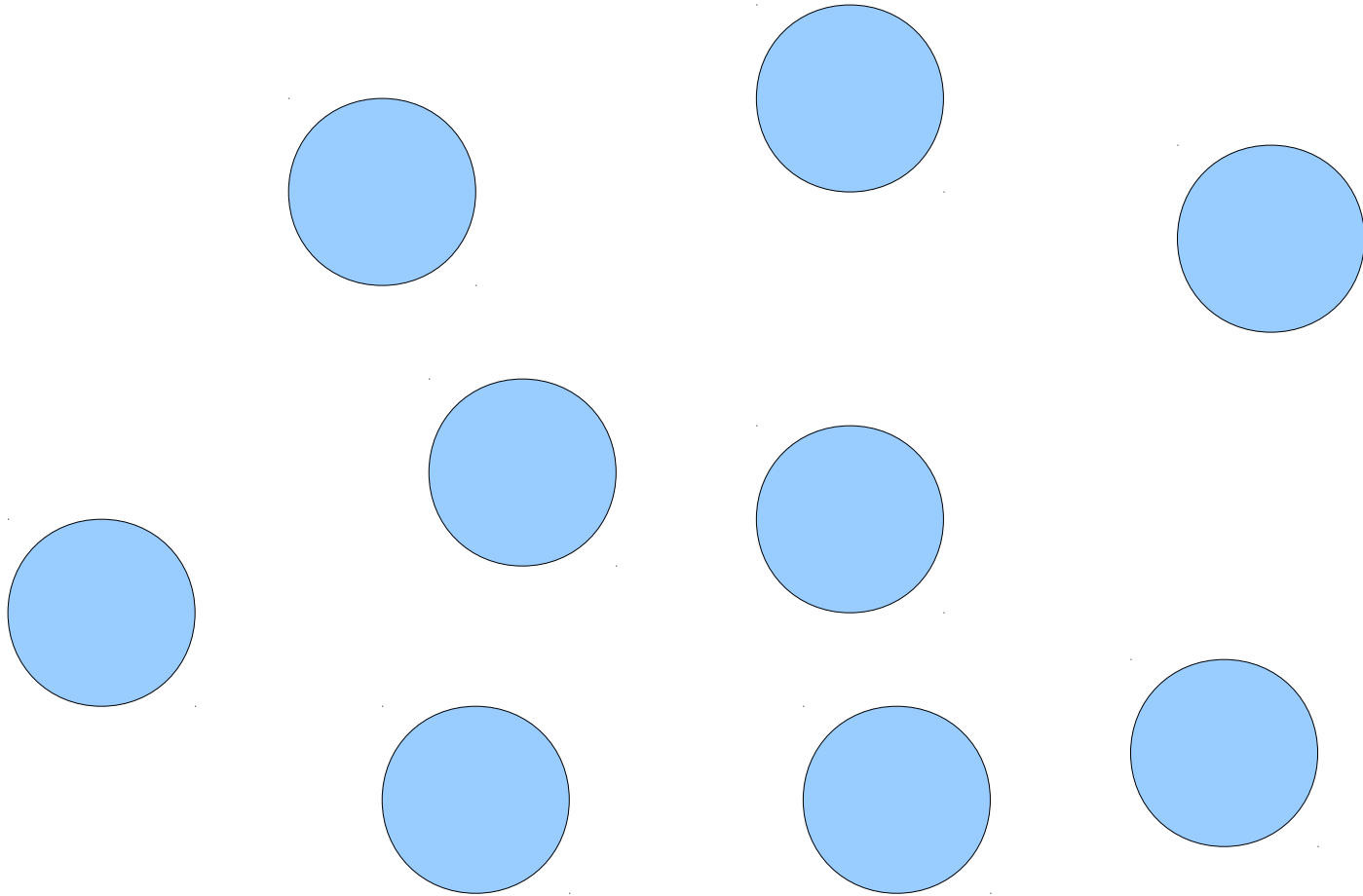
(0, 3) (1, 3) (2, 3) (3, 3)

(0, 4) (1, 4) (2, 4) (3, 4) (4, 4)

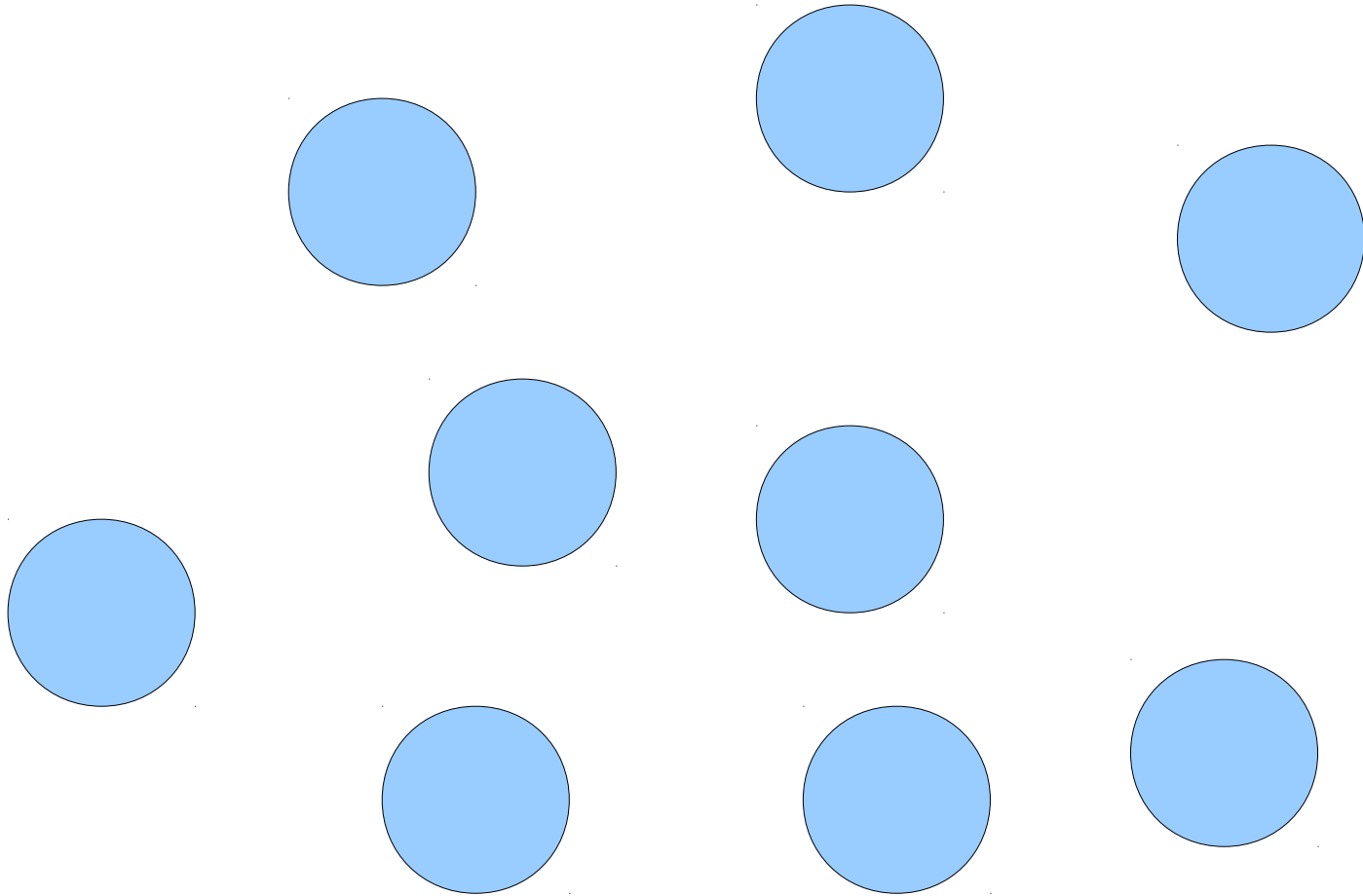
(0, 5) (1, 5) (2, 5) (3, 5) (4, 5) (5, 5)



Generating Combinations

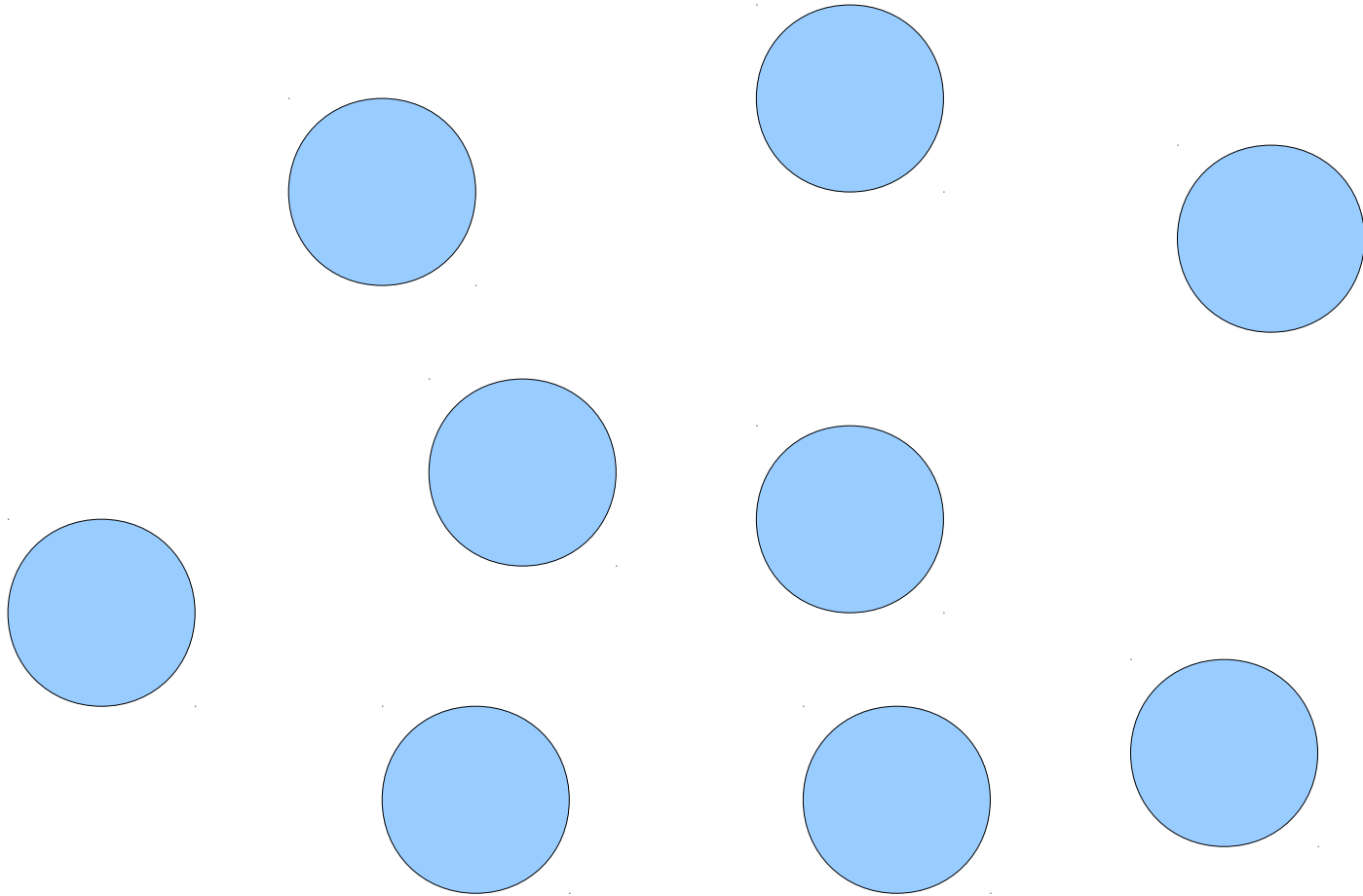


Generating Combinations



How many ways are there to
pick *0* things from this set?

Generating Combinations



How many ways are there to
pick **100** things from this set?

Combinations, Recursively

- How to pick k elements from a set?
- **Base Cases:**
 - If k is 0, the only option is to pick the empty set.
 - Otherwise, if k is greater than the number of elements of the set, there are no options.
- **Recursive Step:**
 - Pick some element x from the set.
 - Find all ways of picking k elements of what remains.
 - Find all ways of picking $k - 1$ elements of what remains, then add x back in.

A Little Word Puzzle

“What nine-letter word can be reduced to a single-letter word one letter at a time by removing letters, leaving it a legal word at each step?”

The Startling Truth

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

The Startling Truth

S	T	A	R	T	I	N	G
---	---	---	---	---	---	---	---

The Startling Truth

S	T	A	R	I	N	G
---	---	---	---	---	---	---

The Startling Truth

S	T	R	I	N	G
---	---	---	---	---	---

The Startling Truth

S	T	I	N	G
---	---	---	---	---

The Startling Truth

S	I	N	G
---	---	---	---

The Startling Truth

S	I	N
---	---	---

The Startling Truth

I	N
---	---

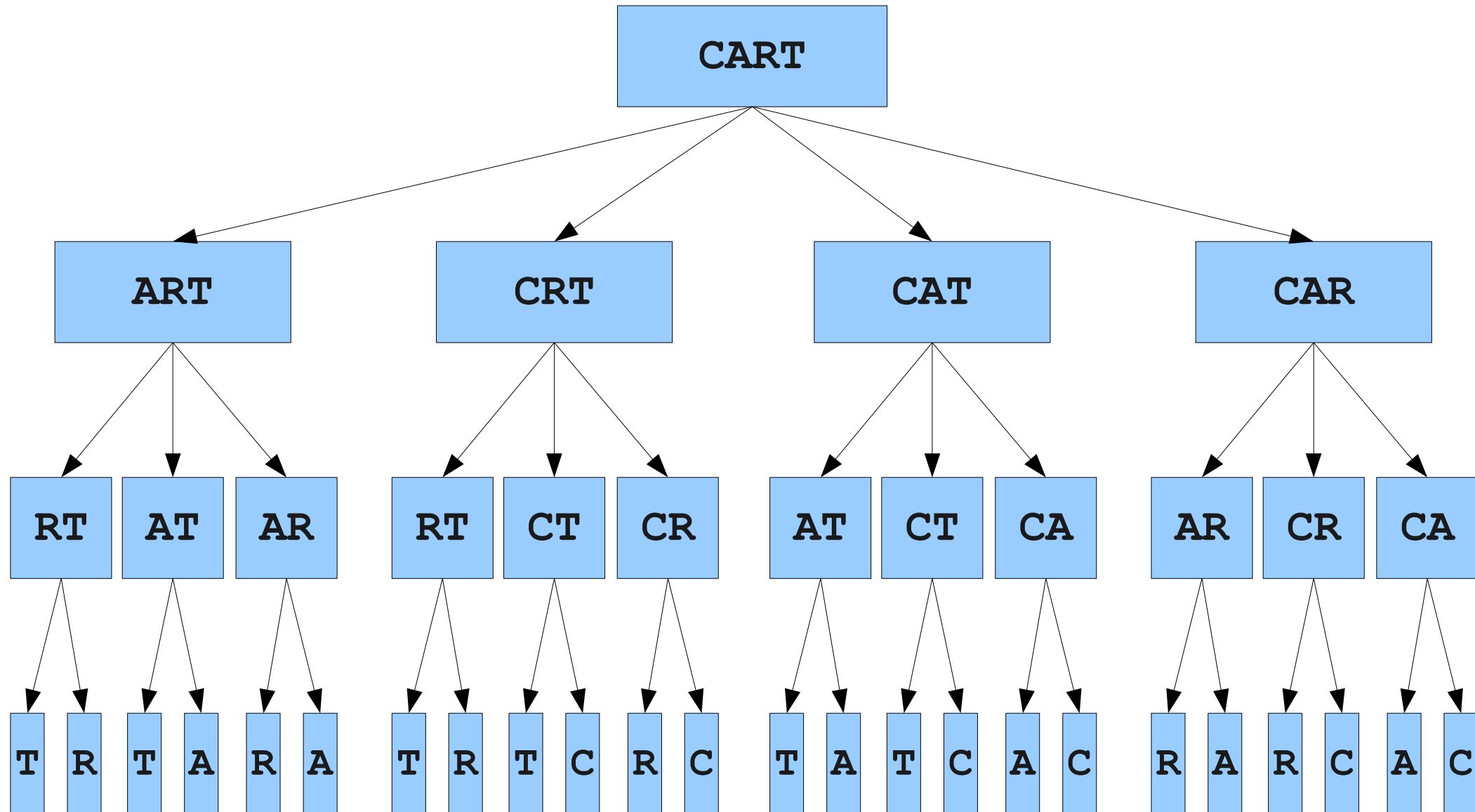
The Startling Truth



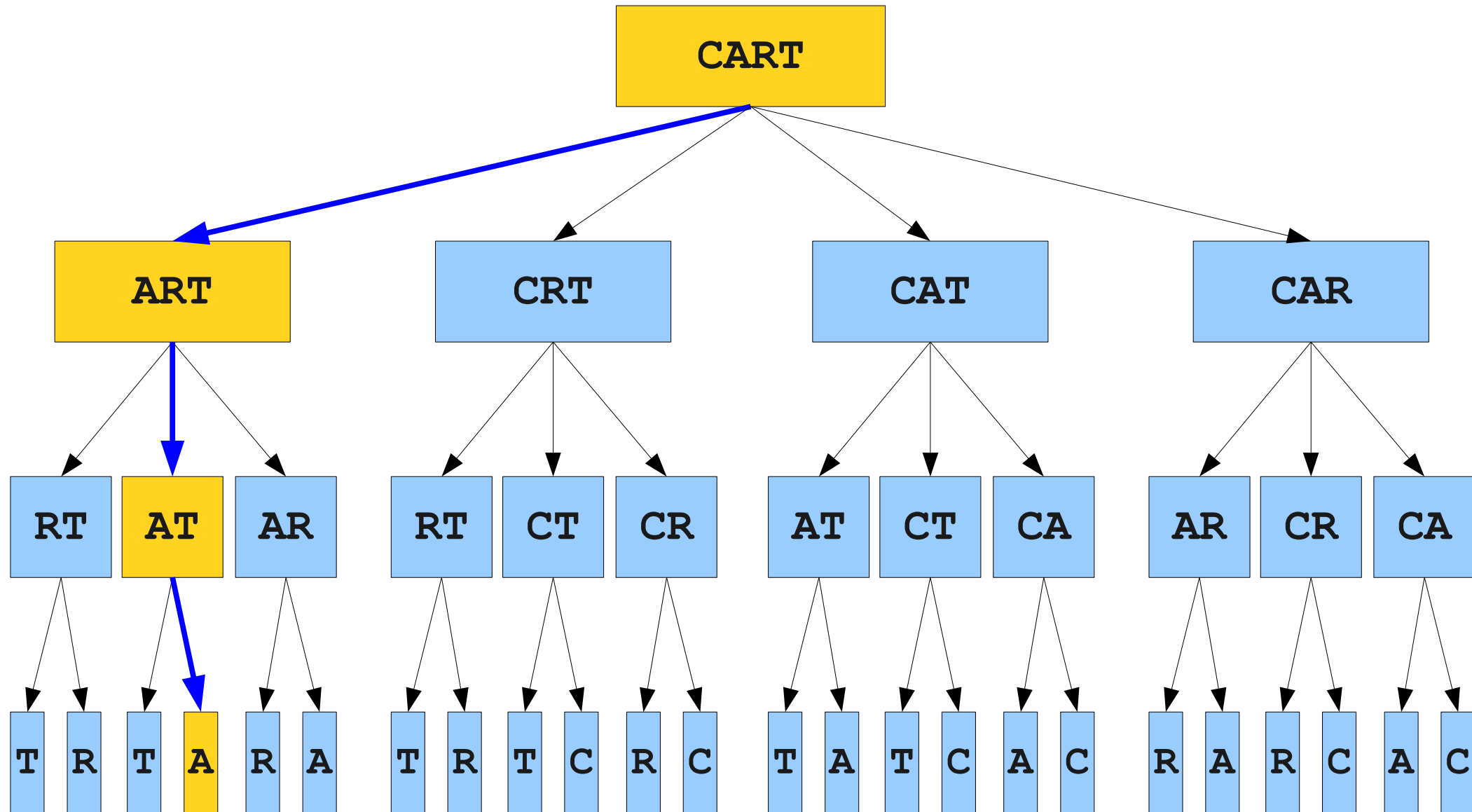
I

Is there **really** just one nine-letter
word with this property?

All Possible Paths



All Possible Paths



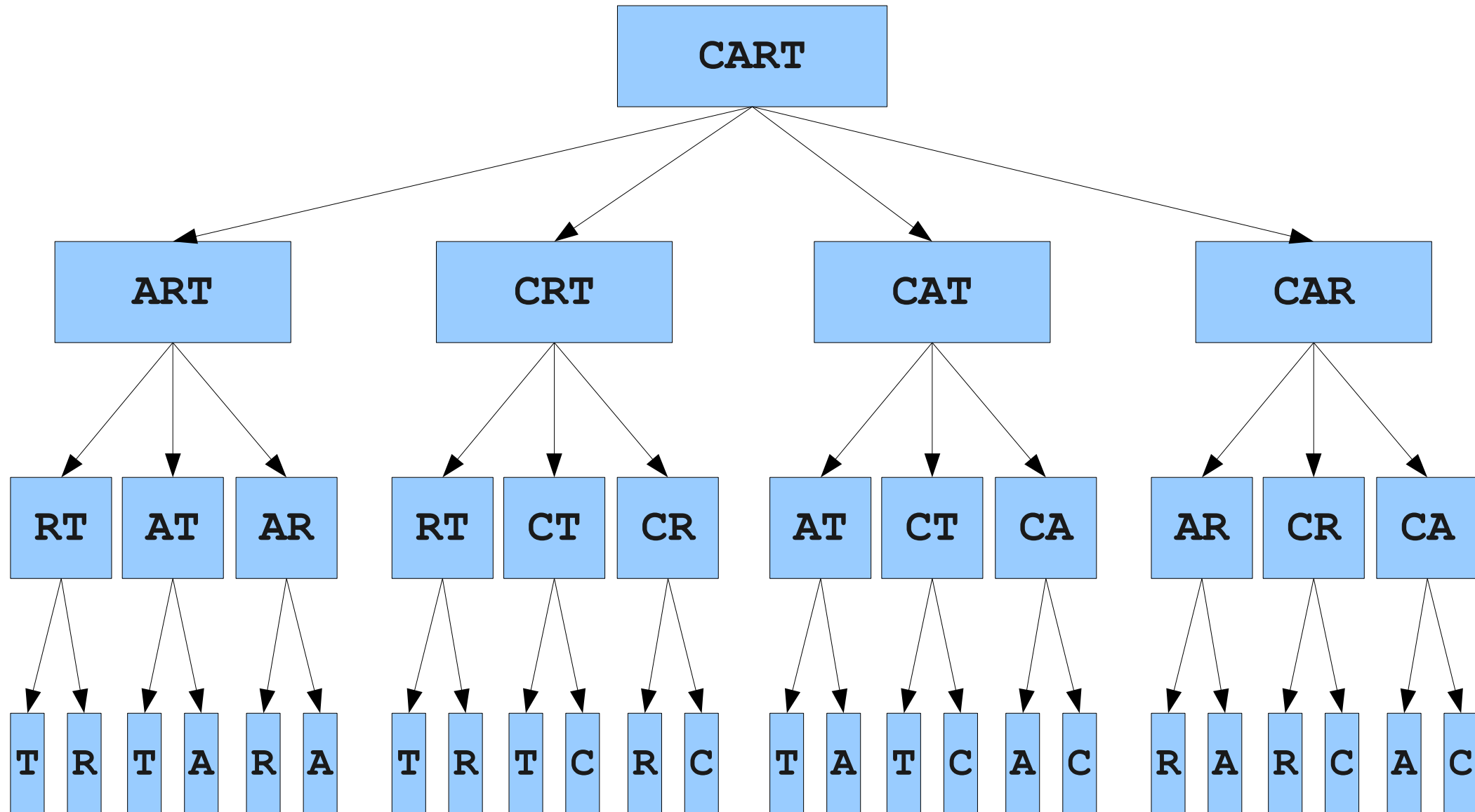
Shrinkable Words

- Let's define a **shrinkable word** as a word that can be reduced down to one letter by removing one character at a time, leaving a word at each step.
- **Base Cases:**
 - Any string that is not a word cannot be a shrinkable word.
 - Any single-letter word is shrinkable.
 - A, I, O
- **Recursive Step:**
 - Any multi-letter word is shrinkable if you can remove a letter to form a shrinkable word.

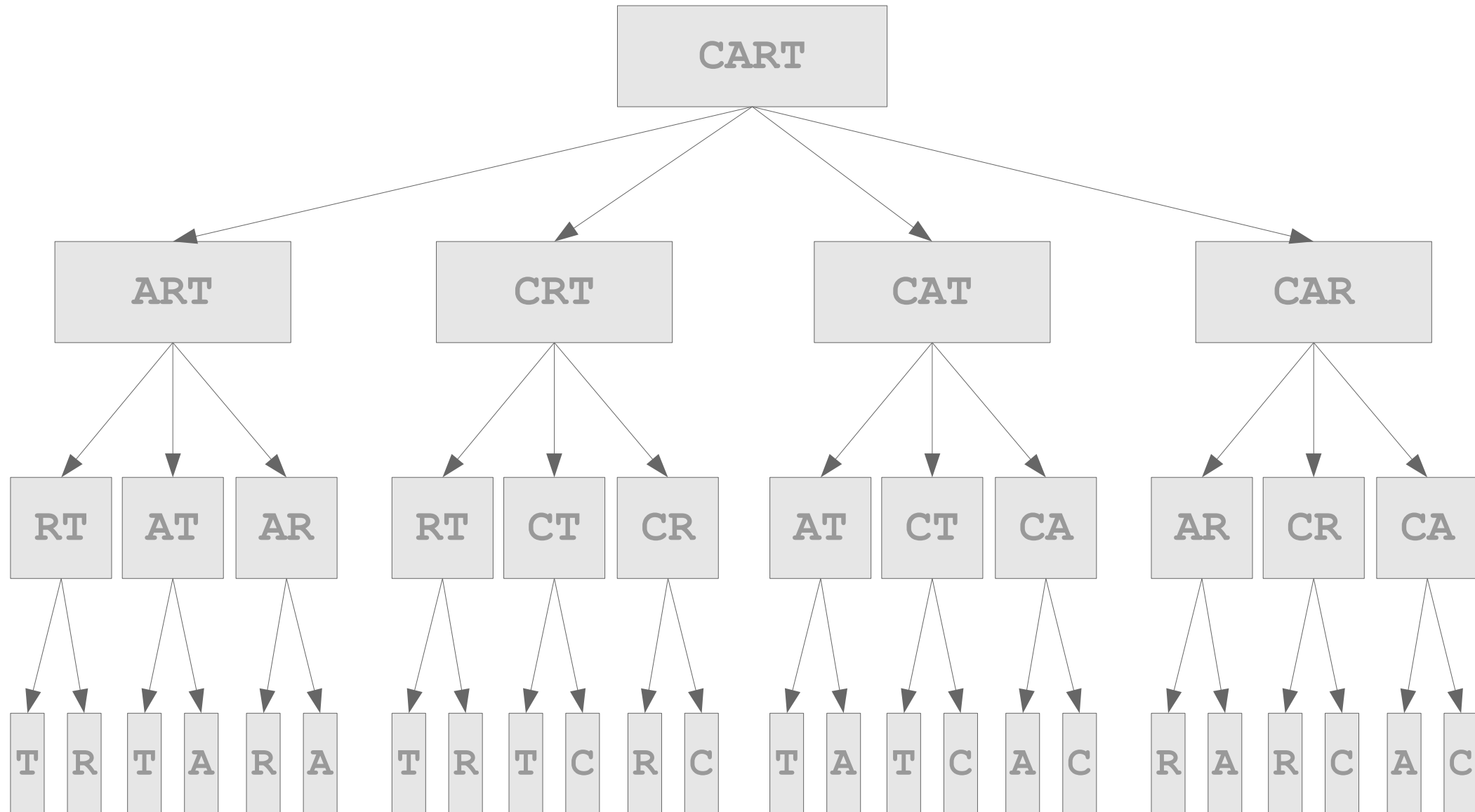
Recursive Backtracking

- The function we have just written is an example of **recursive backtracking**.
- At each step, we try one of many possible options.
- If *any* option succeeds, that's great! We're done.
- If *none* of the options succeed, then this particular problem can't be solved.

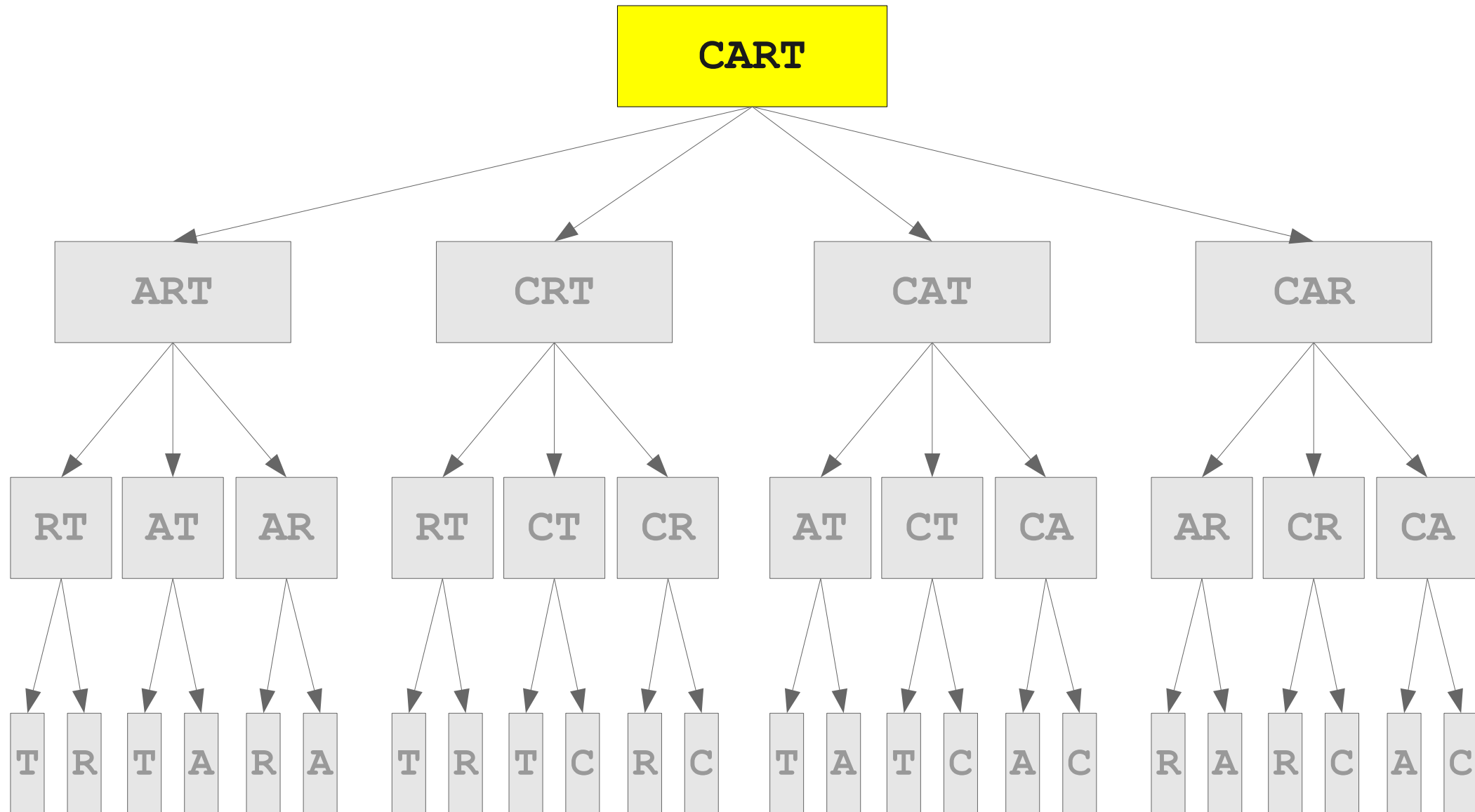
Recursive Backtracking



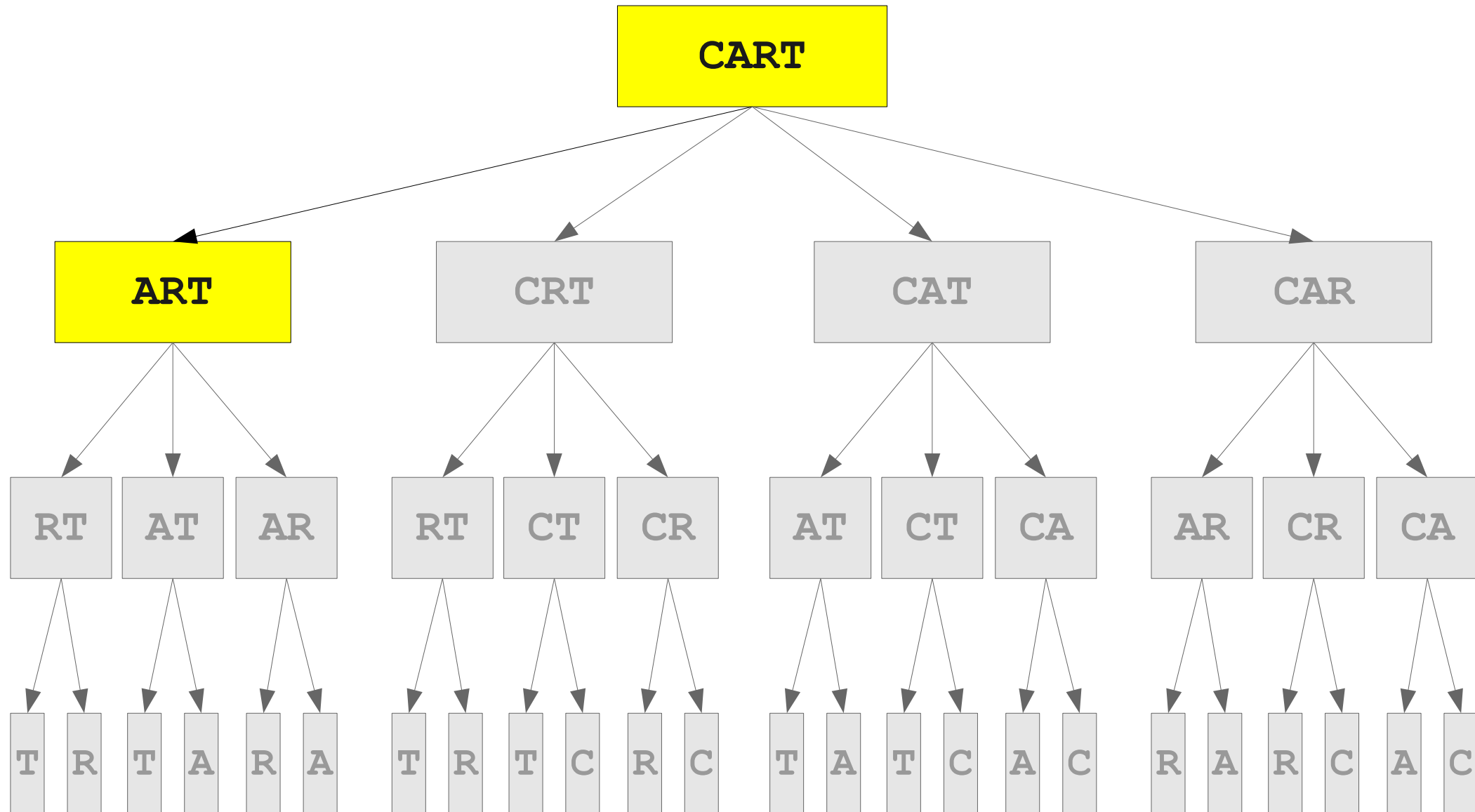
Recursive Backtracking



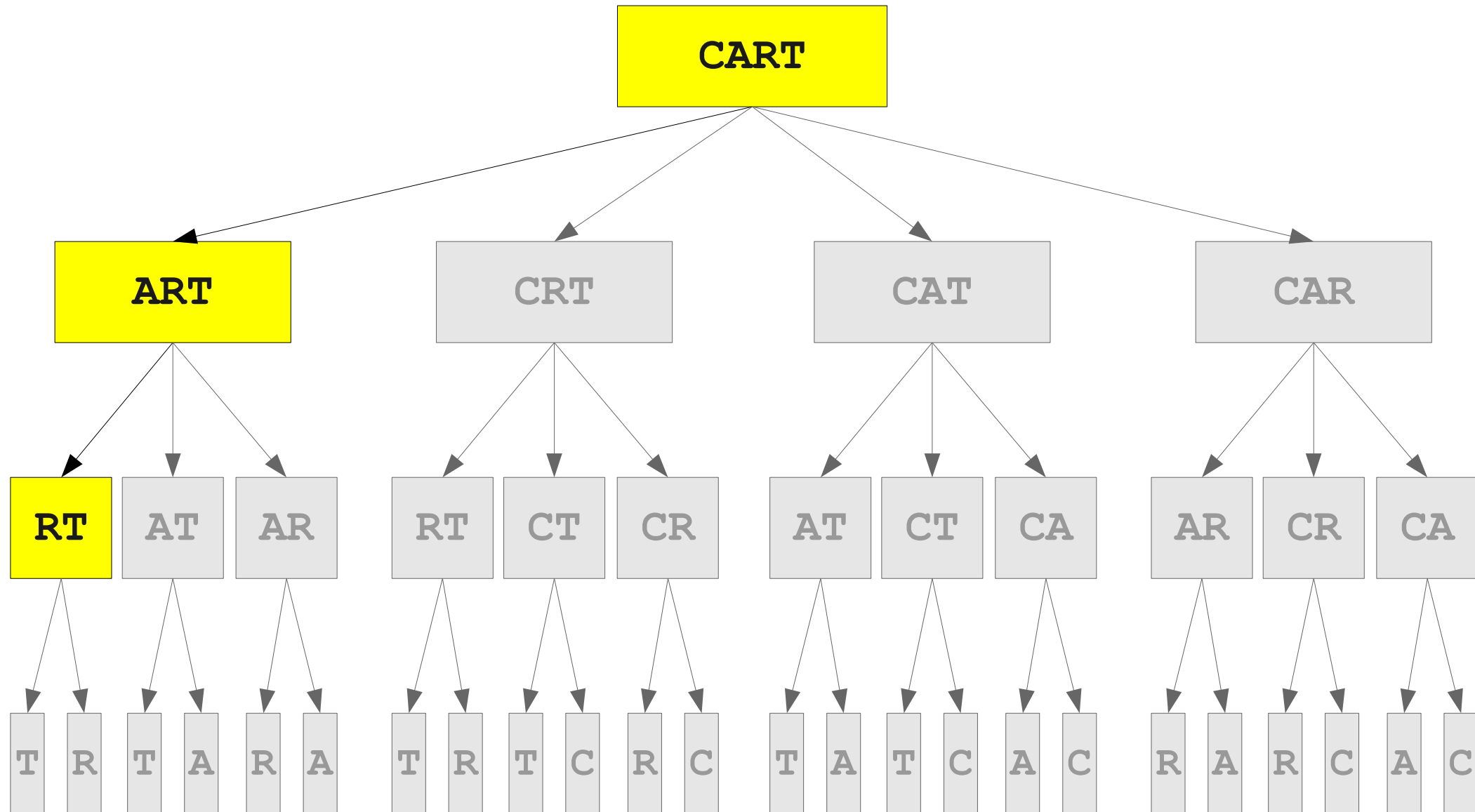
Recursive Backtracking



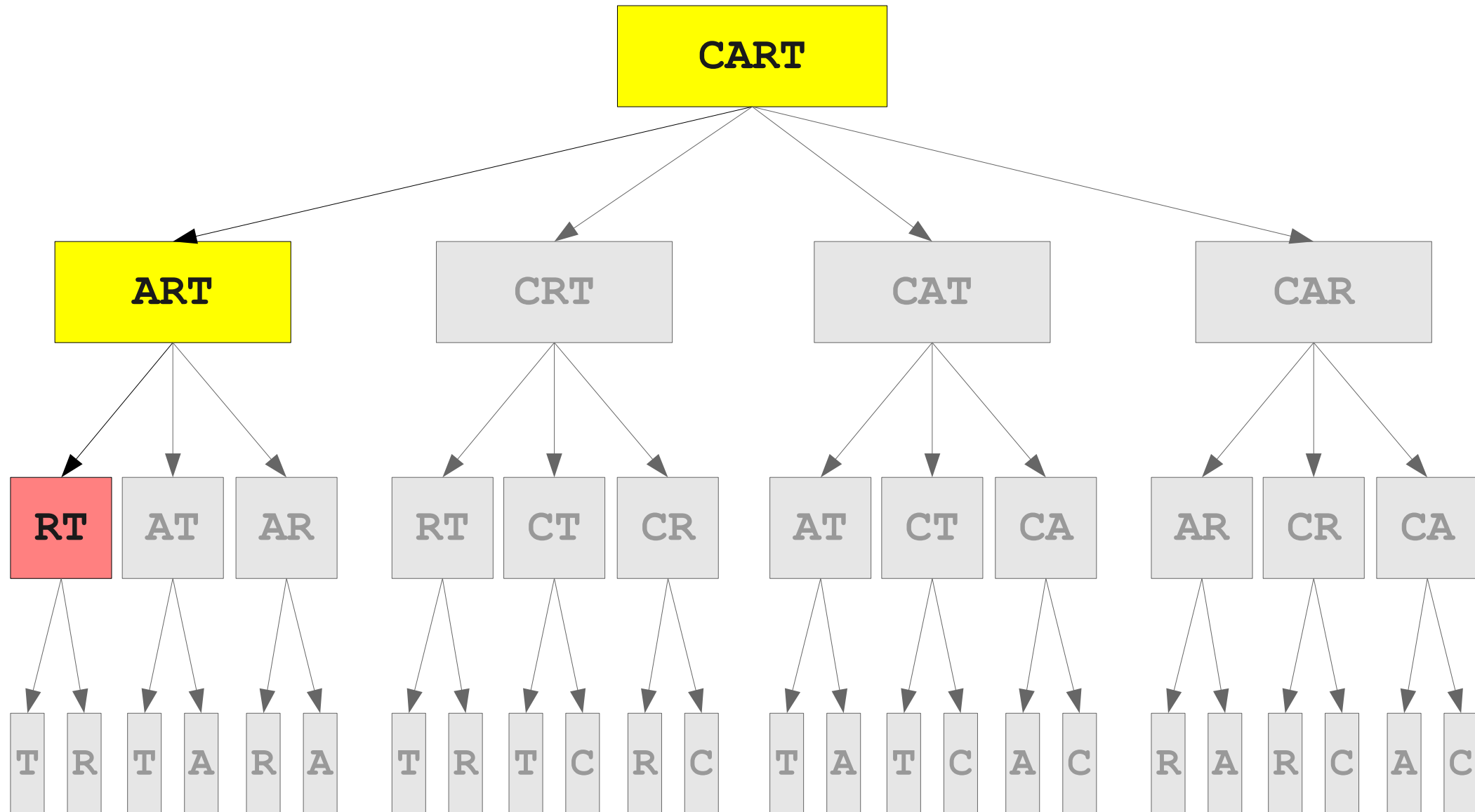
Recursive Backtracking



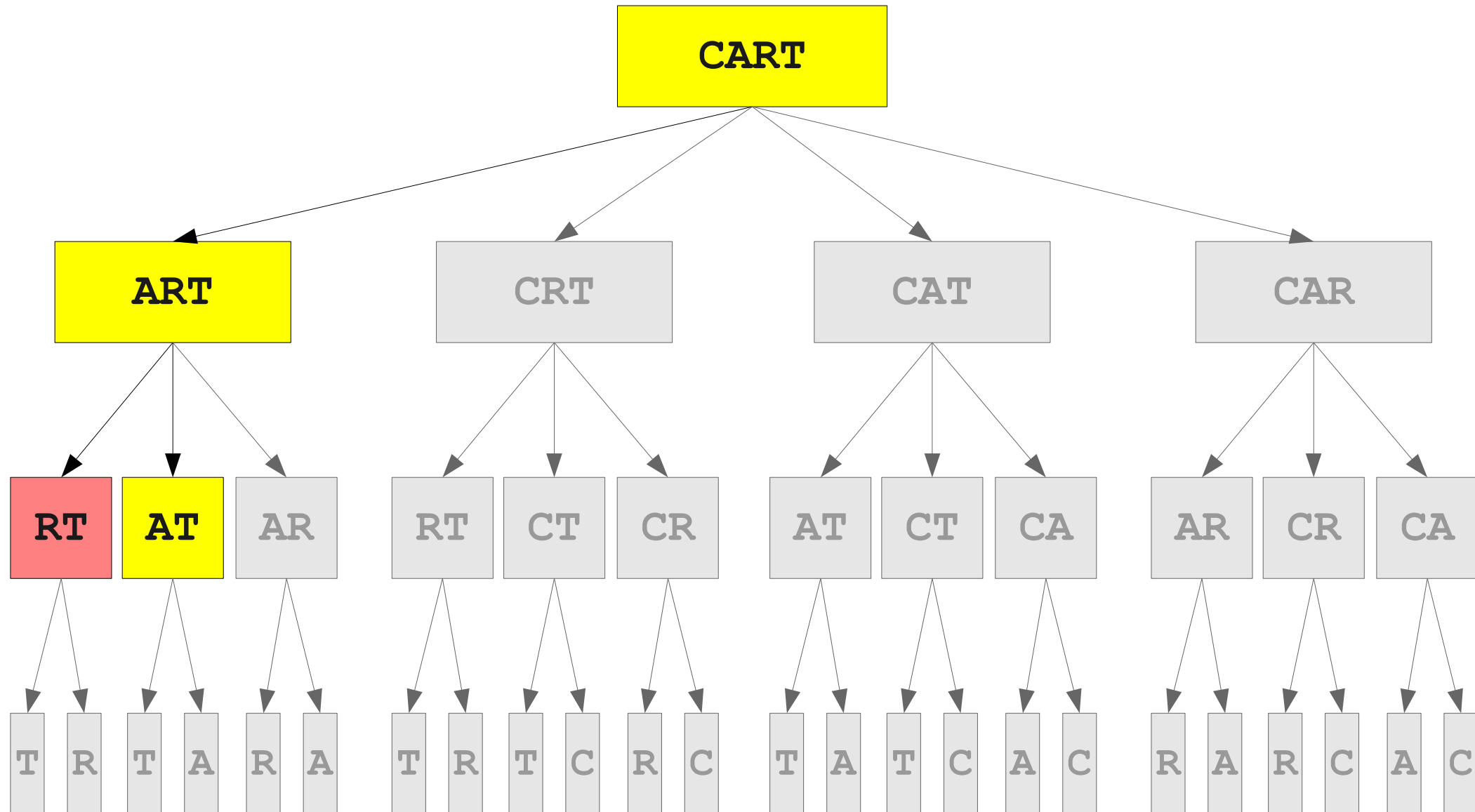
Recursive Backtracking



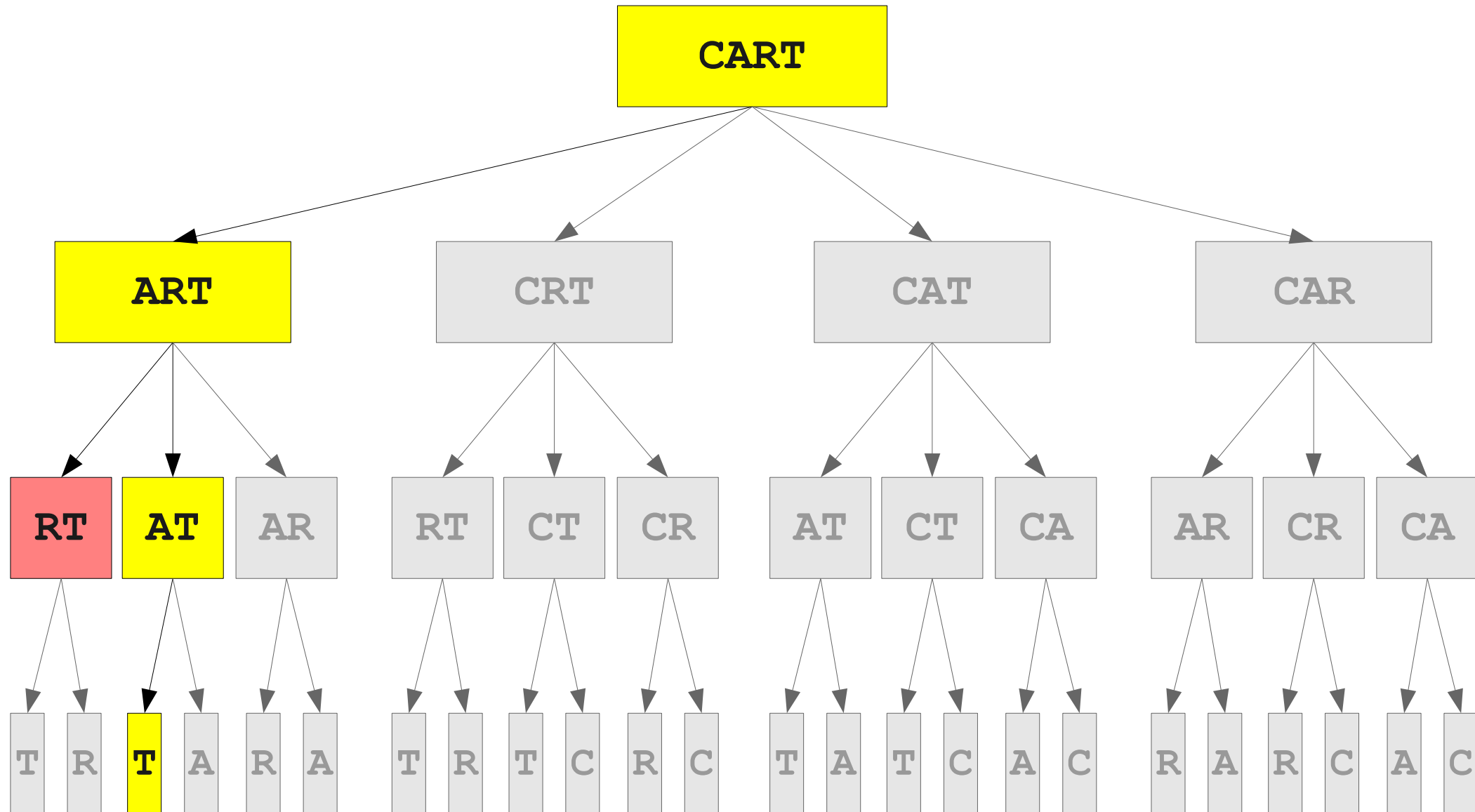
Recursive Backtracking



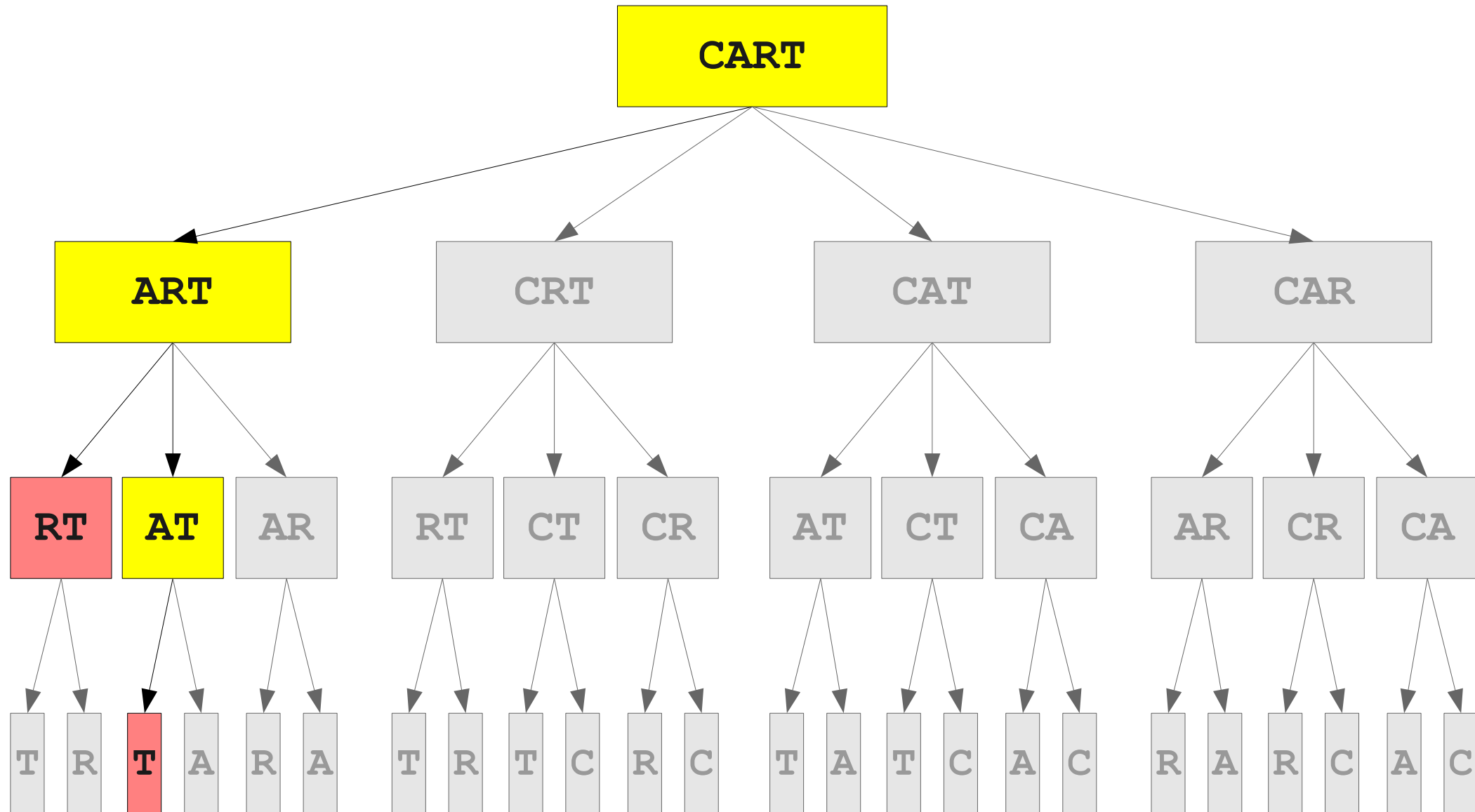
Recursive Backtracking



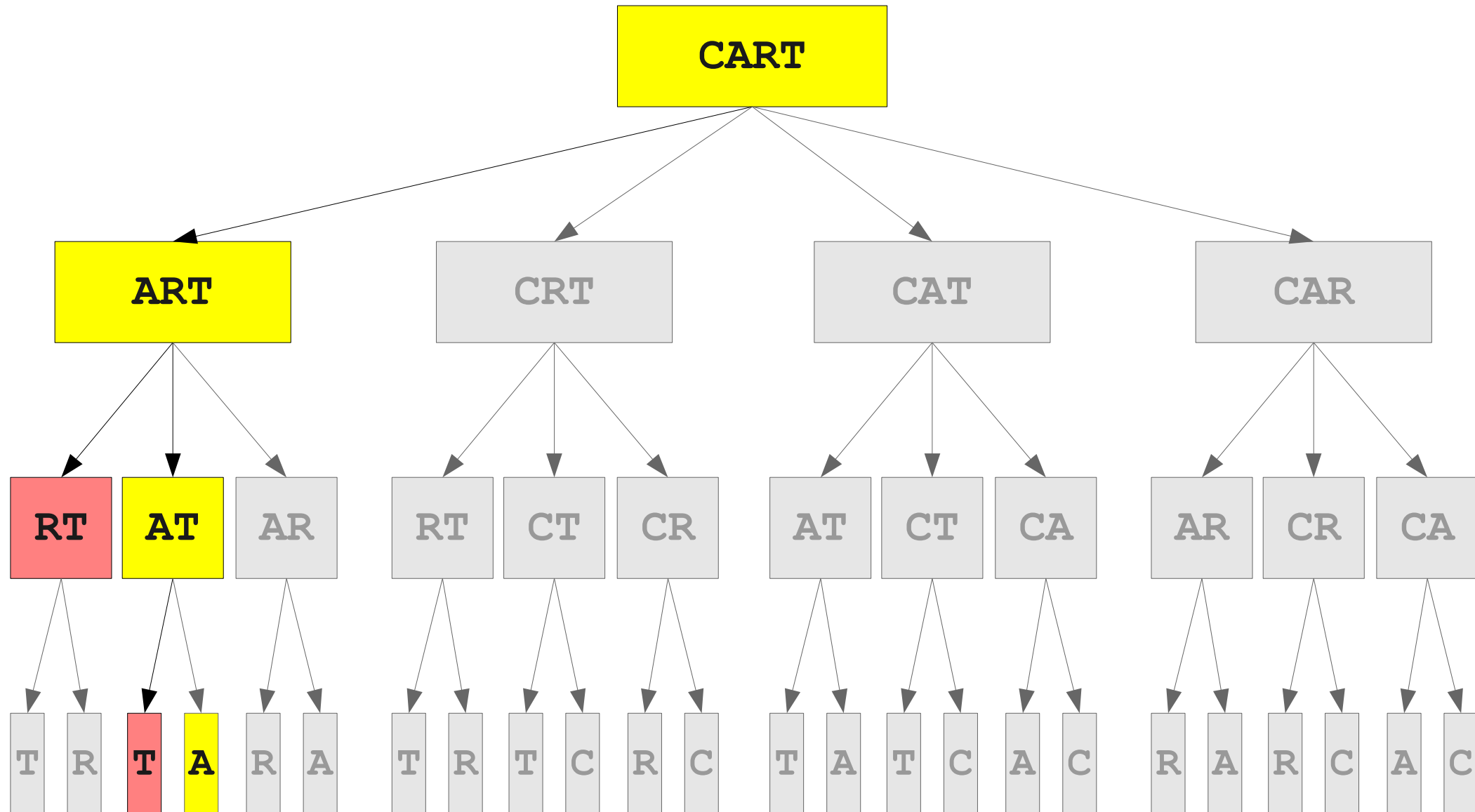
Recursive Backtracking



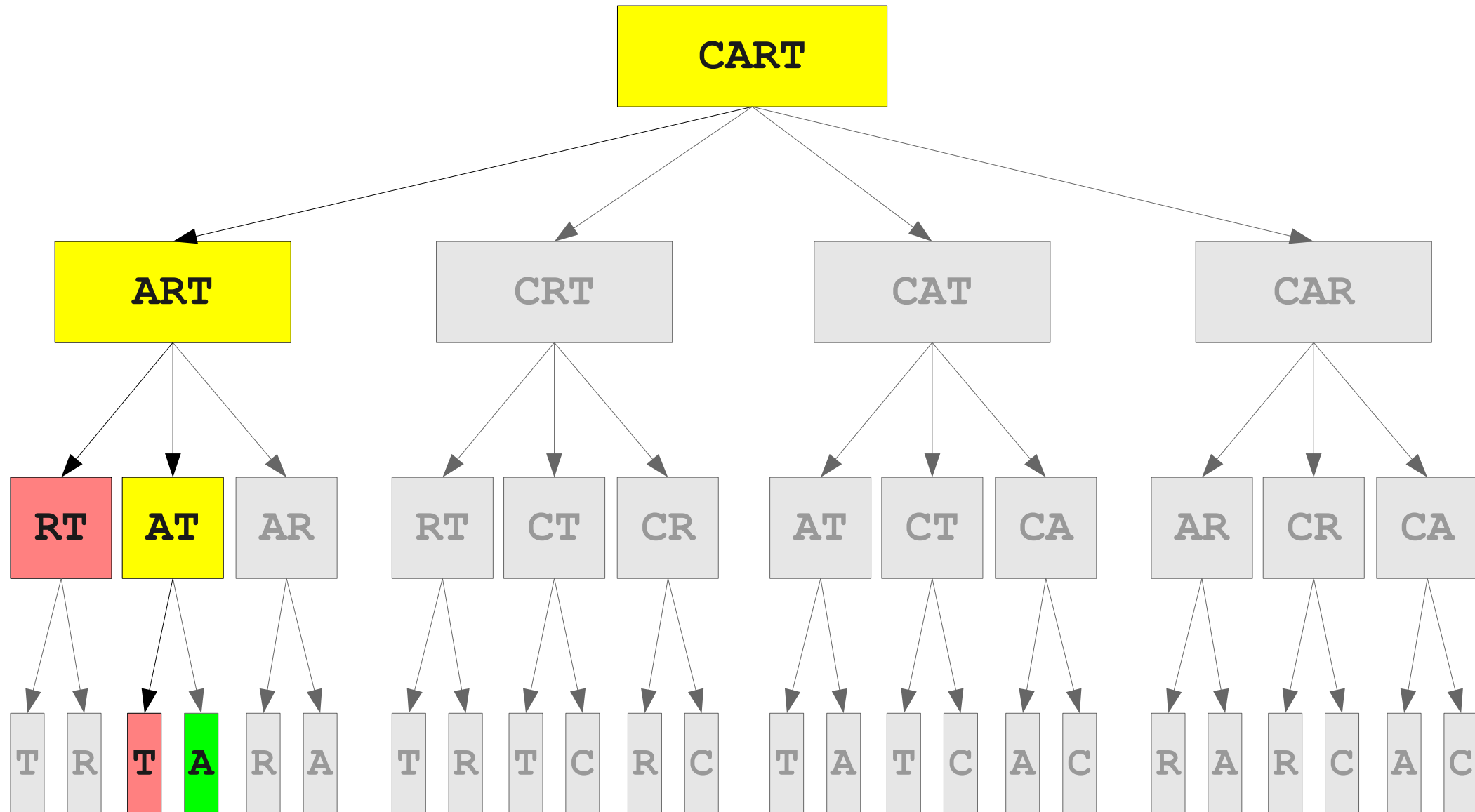
Recursive Backtracking



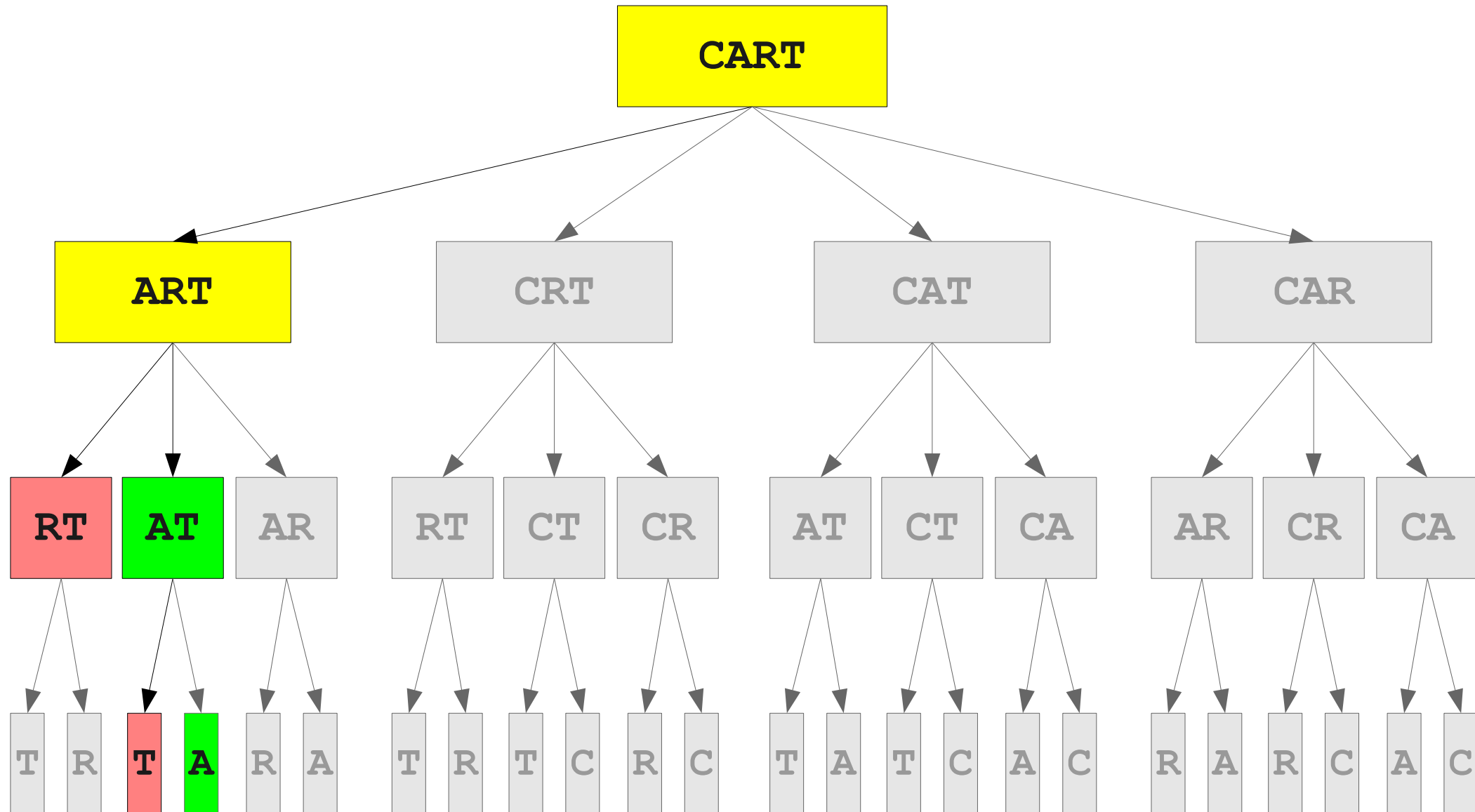
Recursive Backtracking



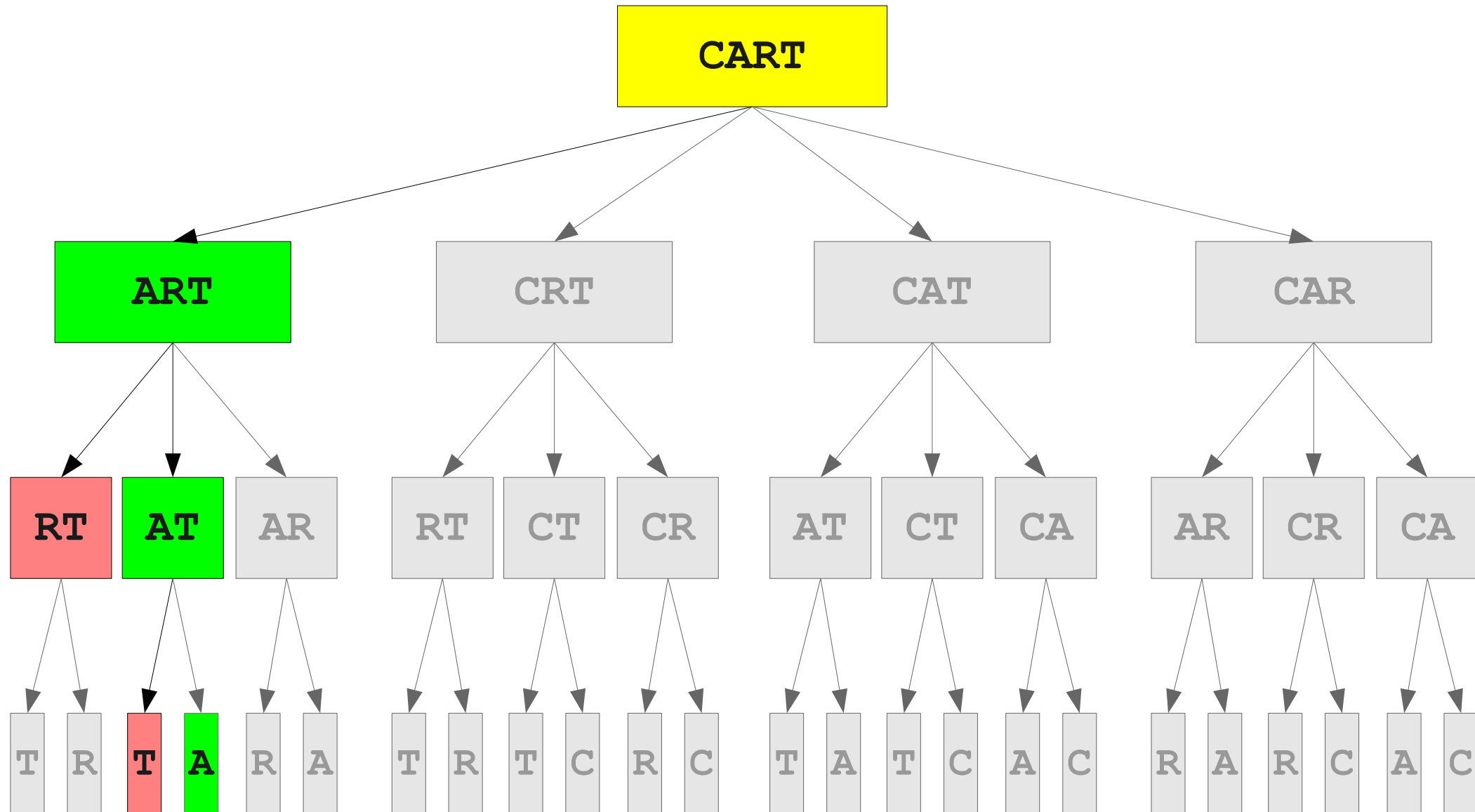
Recursive Backtracking



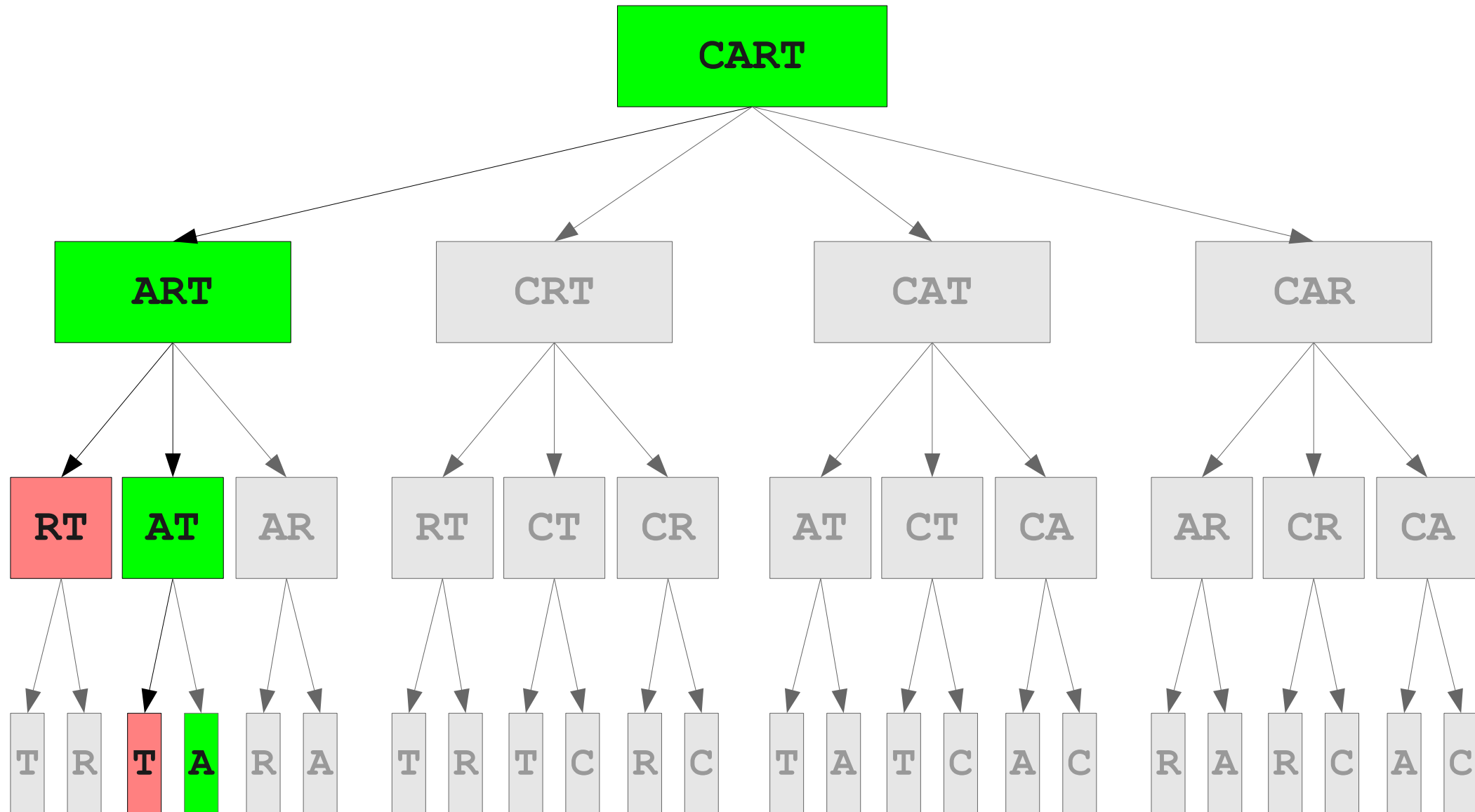
Recursive Backtracking



Recursive Backtracking



Recursive Backtracking



Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

S	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

S	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

S	T	R	T	L	I	N	G
---	---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

S	T	R	T	L	I	N	G
---	---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

S	T	A	T	L	I	N	G
---	---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

S	T	A	T	L	I	N	G
---	---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

S	T	A	R	L	I	N	G
---	---	---	---	---	---	---	---

Failure in Backtracking

S T A R T L I N G

S T A R L I N G



Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

S	T	A	R	L	I	N	G
---	---	---	---	---	---	---	---

T	A	R	L	I	N	G
---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

S	T	A	R	L	I	N	G
---	---	---	---	---	---	---	---

T	A	R	L	I	N	G
---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

S	T	A	R	L	I	N	G
---	---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

S	T	A	R	L	I	N	G
---	---	---	---	---	---	---	---

S	A	R	L	I	N	G
---	---	---	---	---	---	---

Failure in Backtracking

S	T	A	R	T	L	I	N	G
---	---	---	---	---	---	---	---	---

S	T	A	R	L	I	N	G
---	---	---	---	---	---	---	---

S	A	R	L	I	N	G
---	---	---	---	---	---	---

Recursive Backtracking

```
if (problem is sufficiently simple) {  
    return whether or not the problem is solvable  
}  
else {  
    for (each choice) {  
        try out that choice.  
        if (that choice leads to success) {  
            return success  
        }  
    }  
    return failure  
}
```

Recursive Backtracking

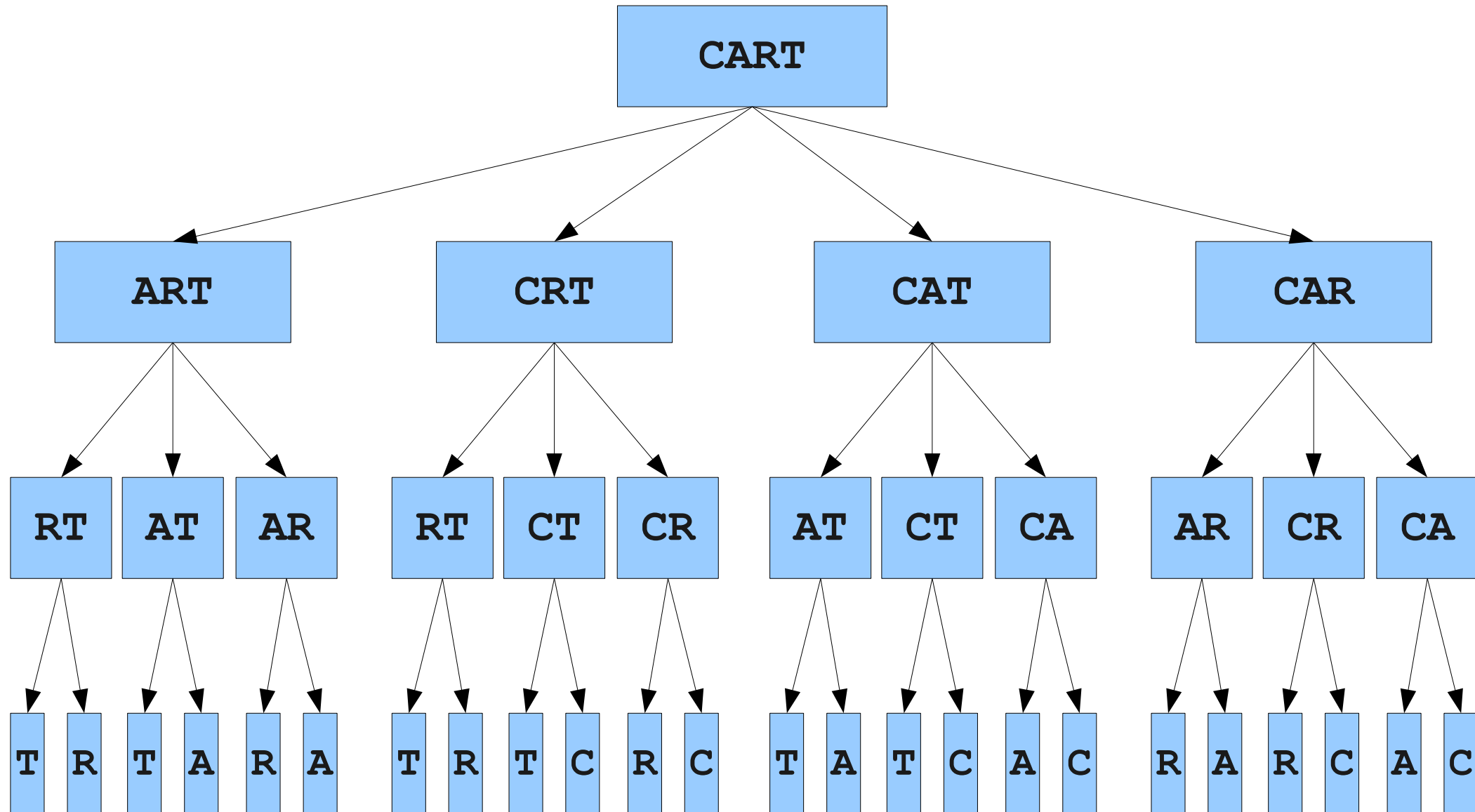
```
if (problem is sufficiently simple) {  
    return whether or not the problem is solvable  
}  
else {  
    for (each choice) {  
        try out that choice.  
        if (that choice leads to success) {  
            return success  
        }  
    }  
    return failure  
}
```

Note that if it succeeds, then we return success. If it doesn't succeed, that doesn't mean we've failed – it just means we need to try out the next option.

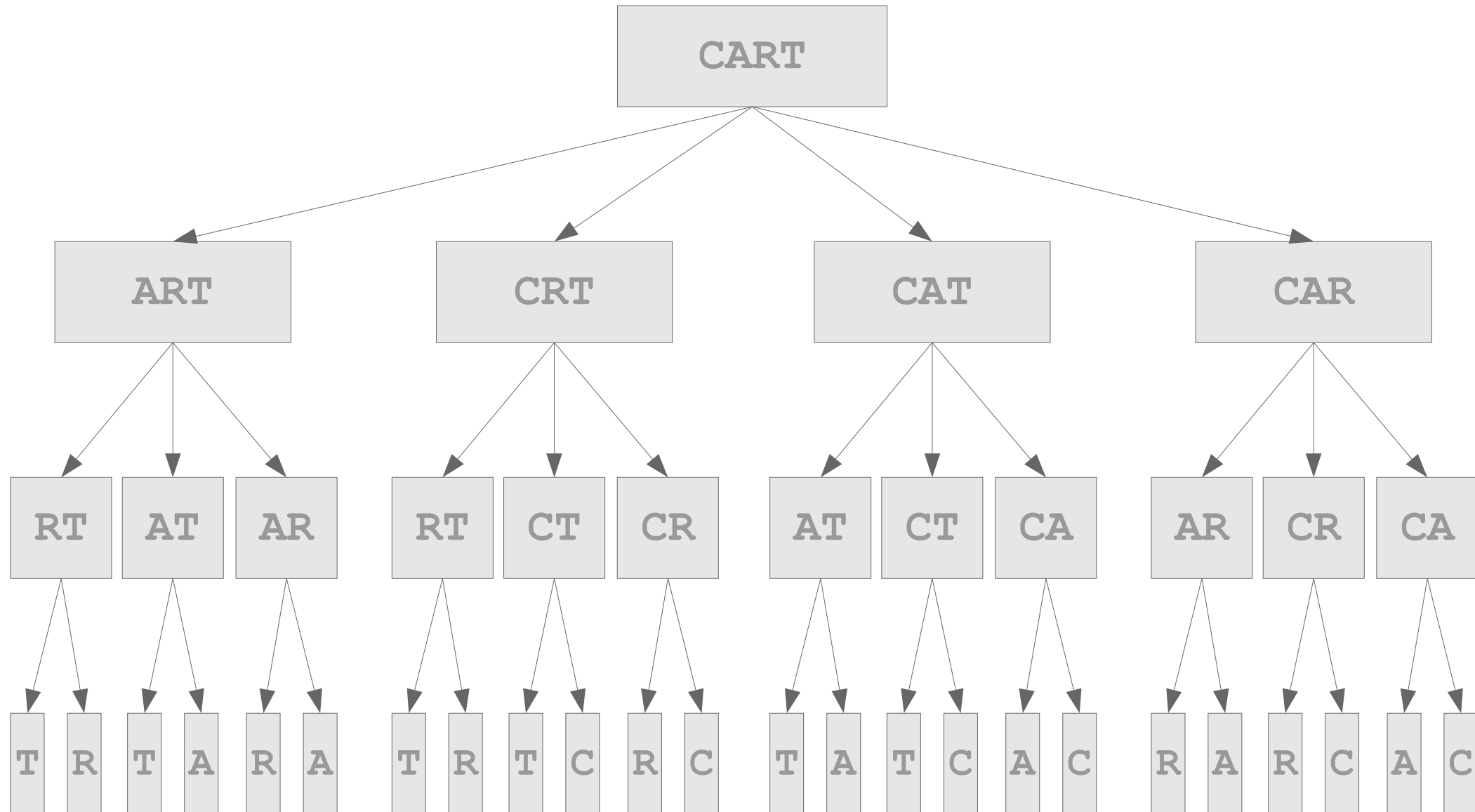
Failure in Backtracking

- Returning false in recursive backtracking does ***not*** mean that the entire problem is unsolvable!
- Instead, it just means that the current subproblem is unsolvable.
- Whoever made the call to this function can then try other options.
- Only when all options are exhausted can we know that the problem is unsolvable.

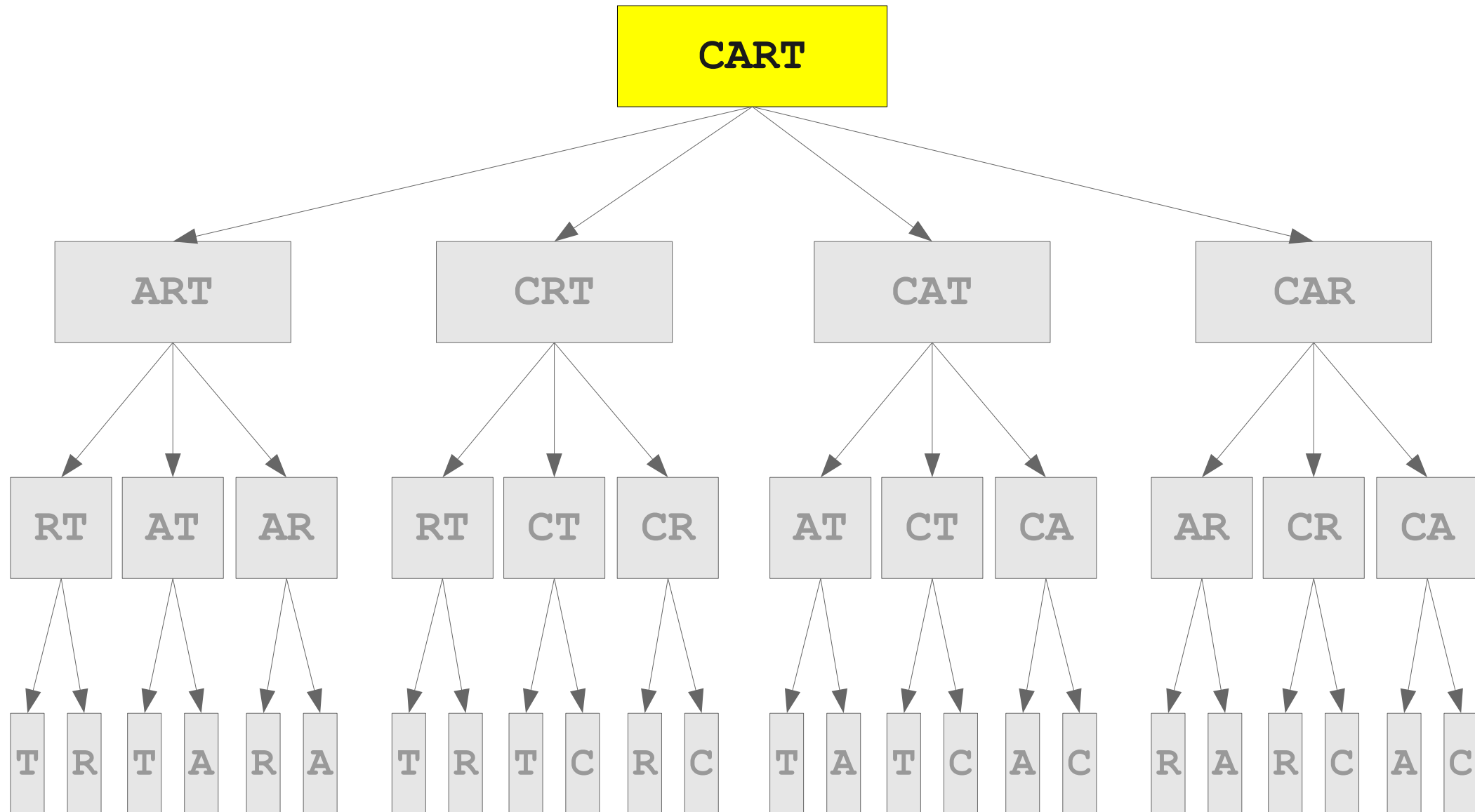
Ur Doin It Rong!



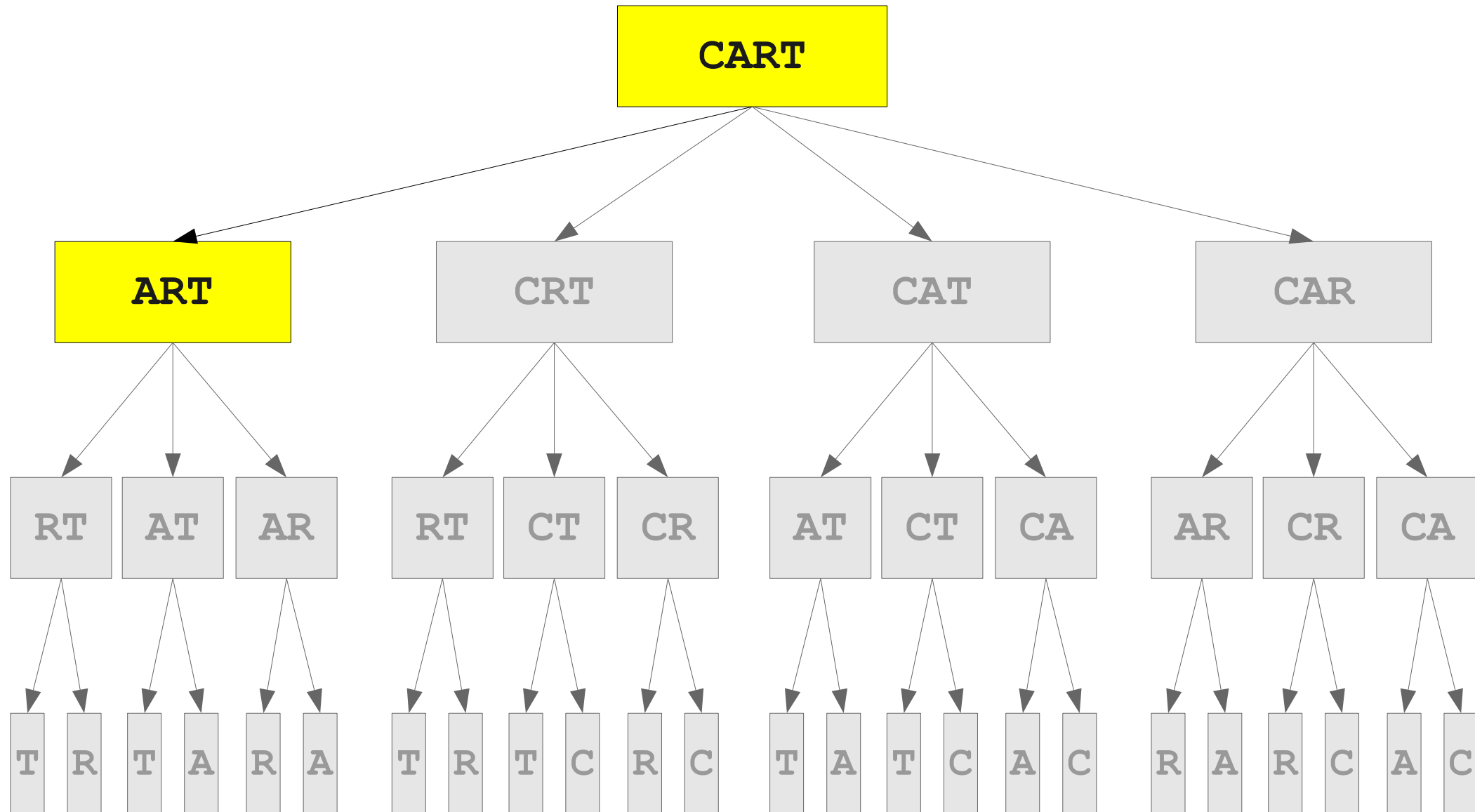
Ur Doin It Rong!



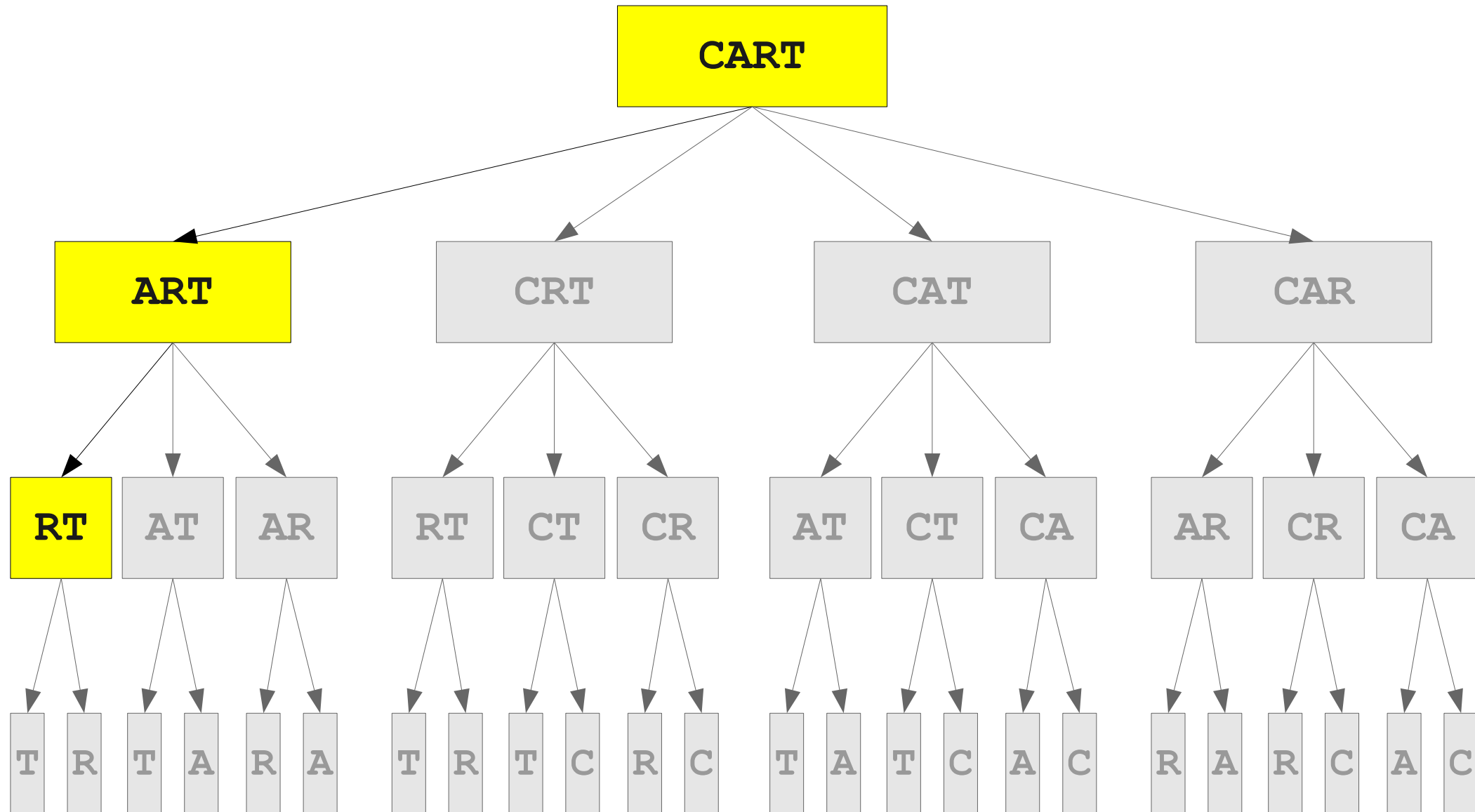
Ur Doin It Rong!



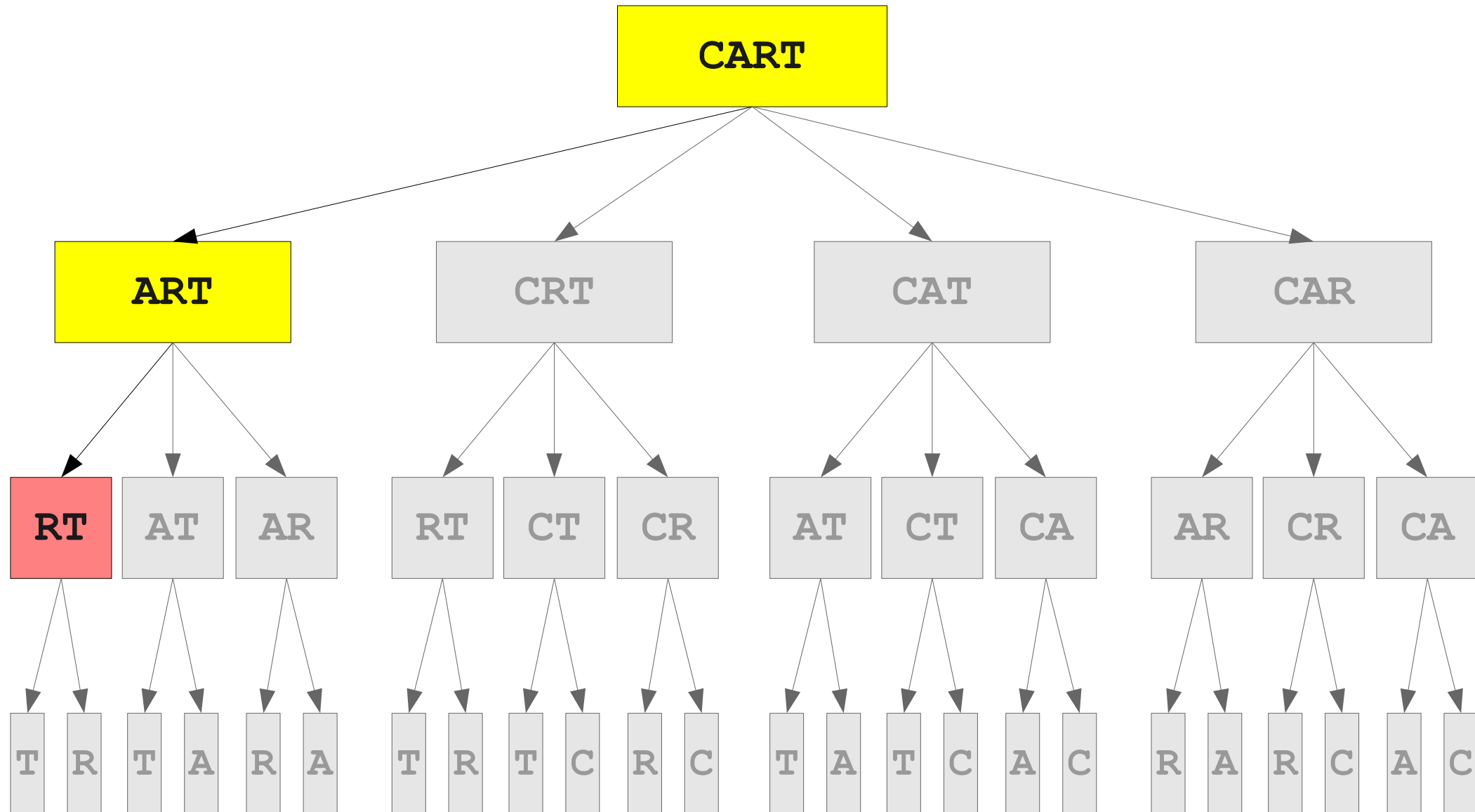
Ur Doin It Rong!



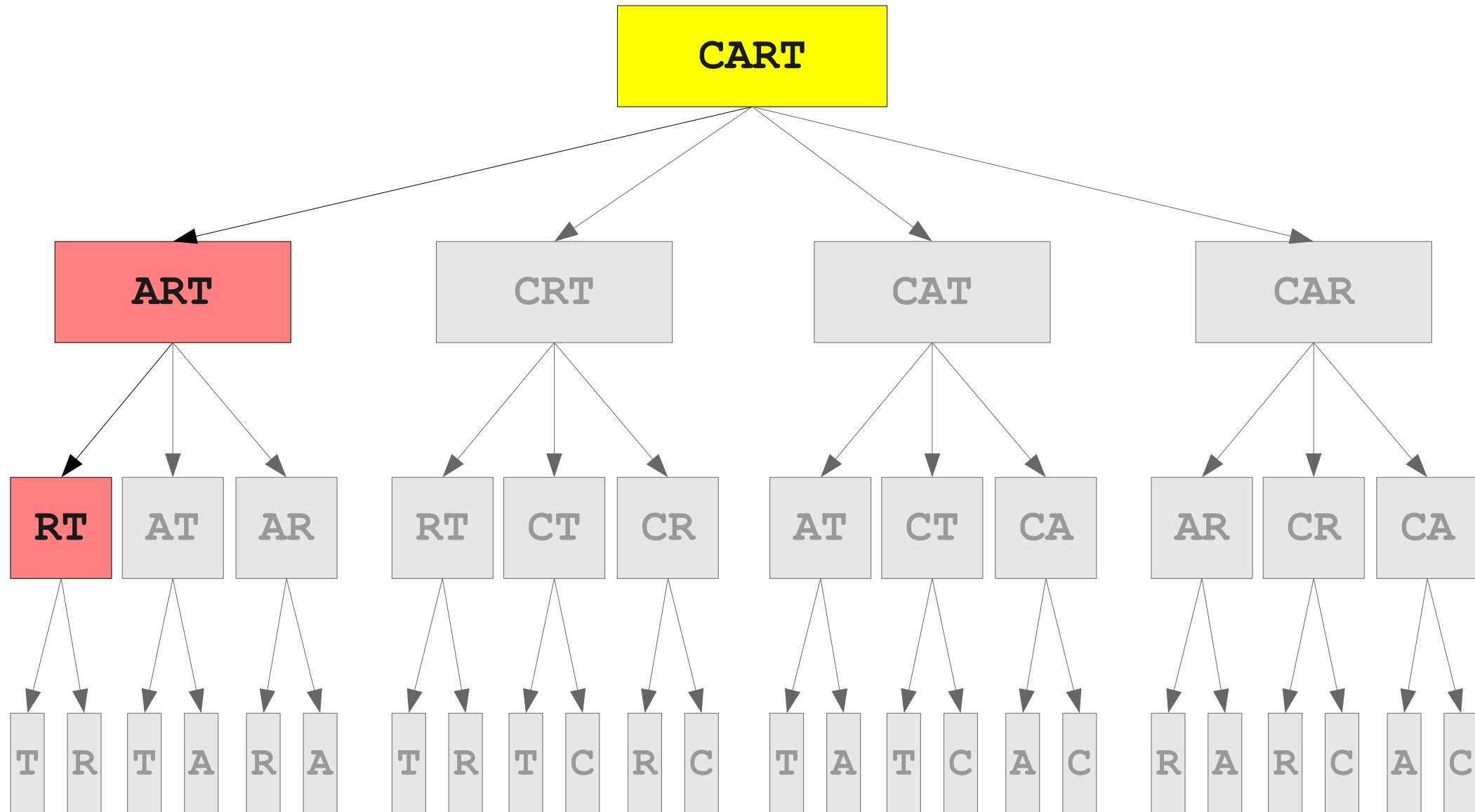
Ur Doin It Rong!



Ur Doin It Rong!



Ur Doin It Rong!



Ur Doin It Rong!

