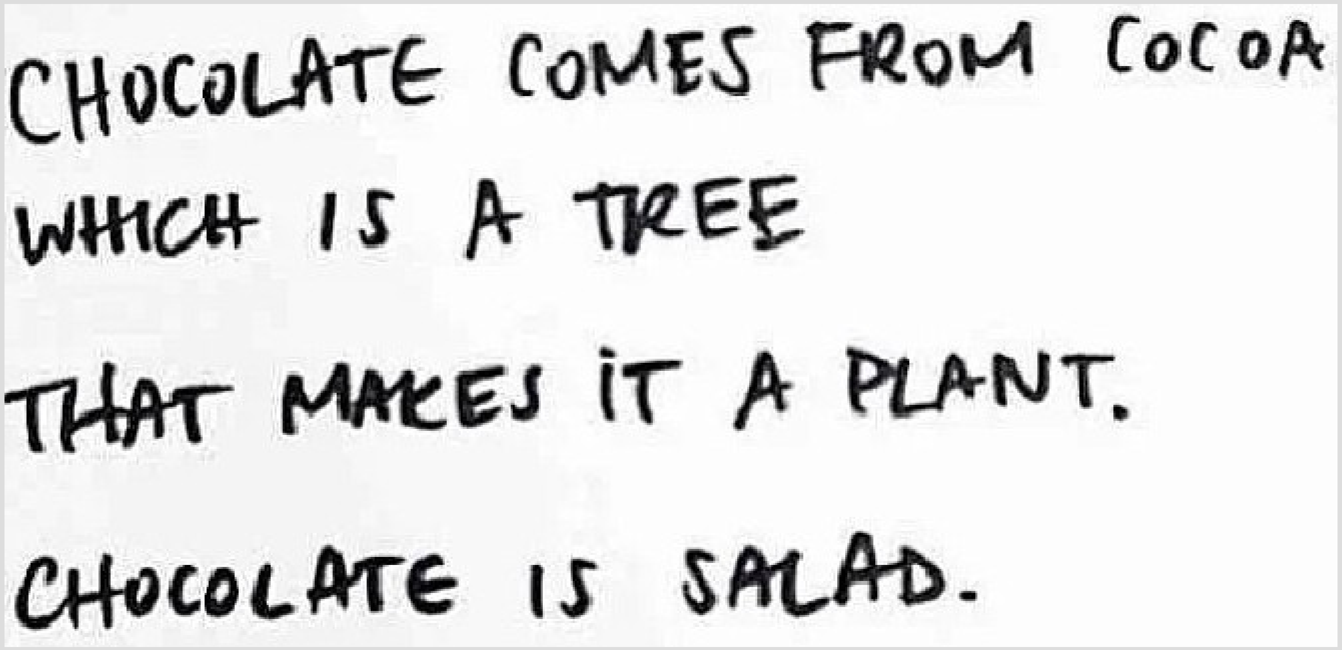
Using first order logic (Ch. 9)


CHOCOLATE COMES FROM COCOA
WHICH IS A TREE
THAT MAKES IT A PLANT.
CHOCOLATE IS SALAD.

Backward chaining

Backward chaining is almost the opposite of forward chaining (and eliminating irrelevancy)

You try all sentences that are of the form: $P1 \land P2 \land ... \Rightarrow Goal$, and try to find a way to satisfy P1, P2, ... recursively

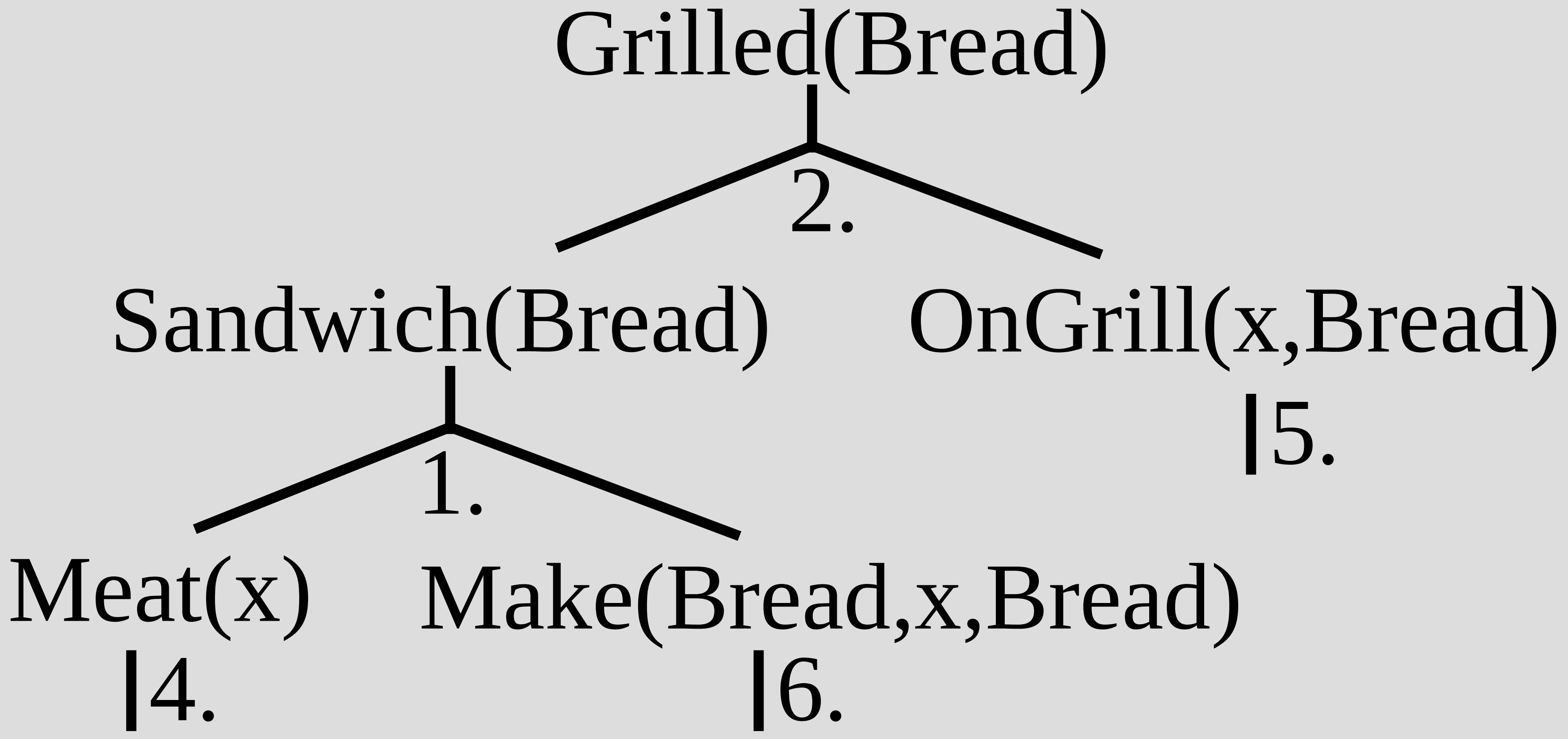
This is similar to a depth first search AND/OR trees (OR are possible substitutions while AND nodes are the sentence conjunctions)

Backward chaining

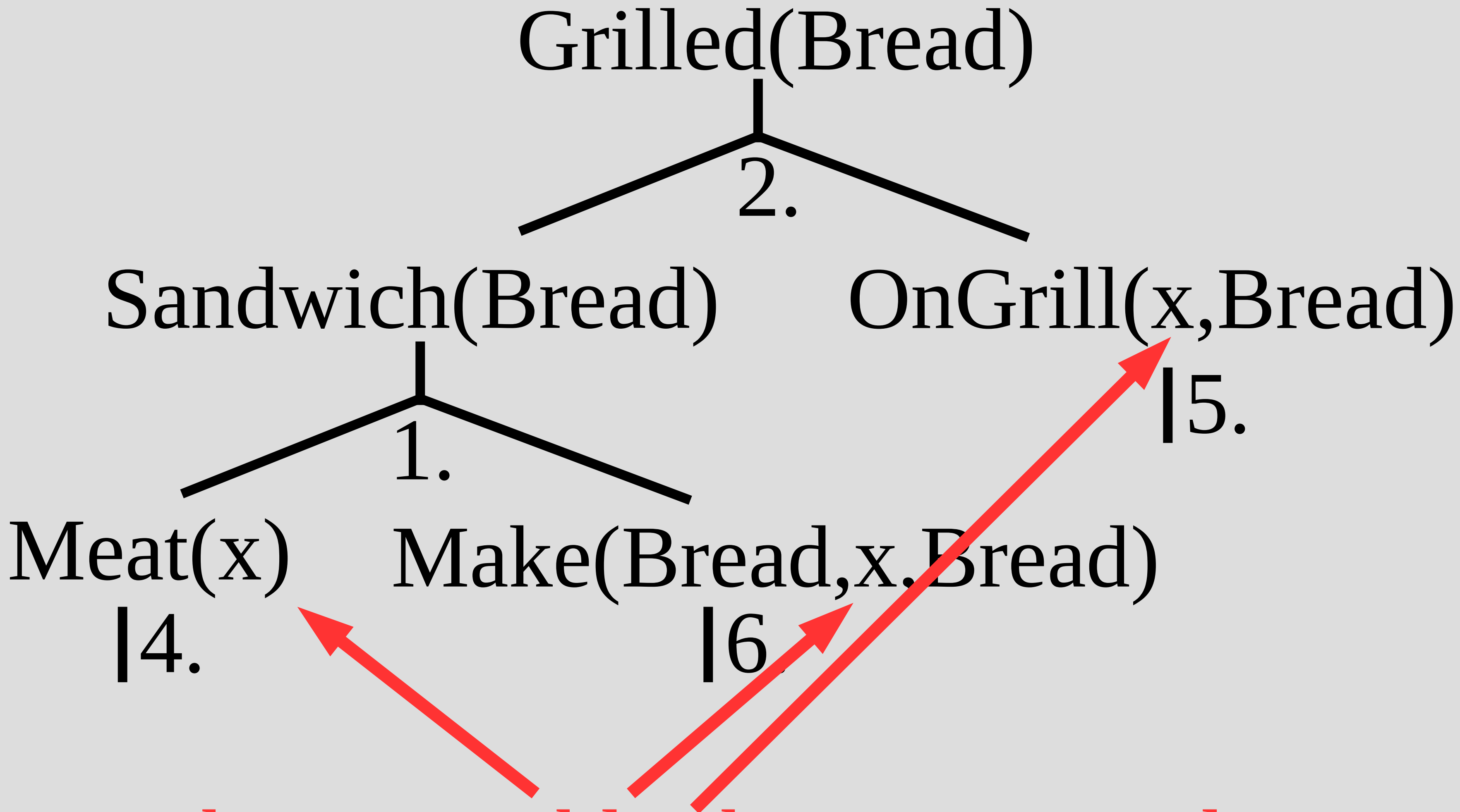
Let's go back to this and backward chain Grilled(Bread)

1. $\forall x\ Meat(x) \land Make(Bread, x, Bread) \Rightarrow Sandwich(Bread)$
2. $\forall x, y\ OnGrill(x, y) \land Sandwich(y) \Rightarrow Grilled(y)$
3. $\forall x, y\ OnGrill(x, y) \land Meat(y) \Rightarrow Grilled(y)$
4. $\exists x\ Meat(x)$
5. $\forall x, y\ OnGrill(x, y)$
6. $\forall x, y, z\ Make(x, y, z)$

Backward chaining

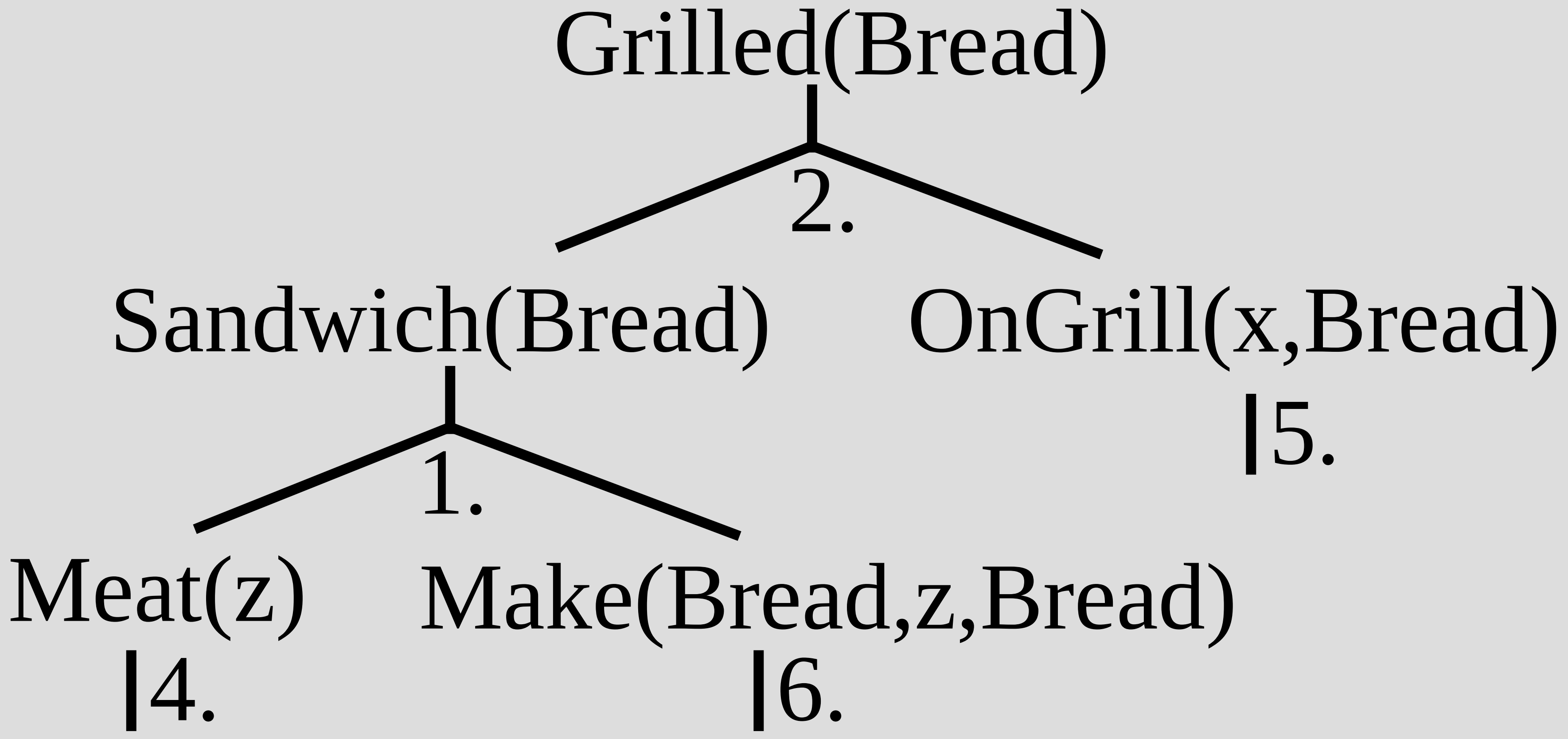


Backward chaining



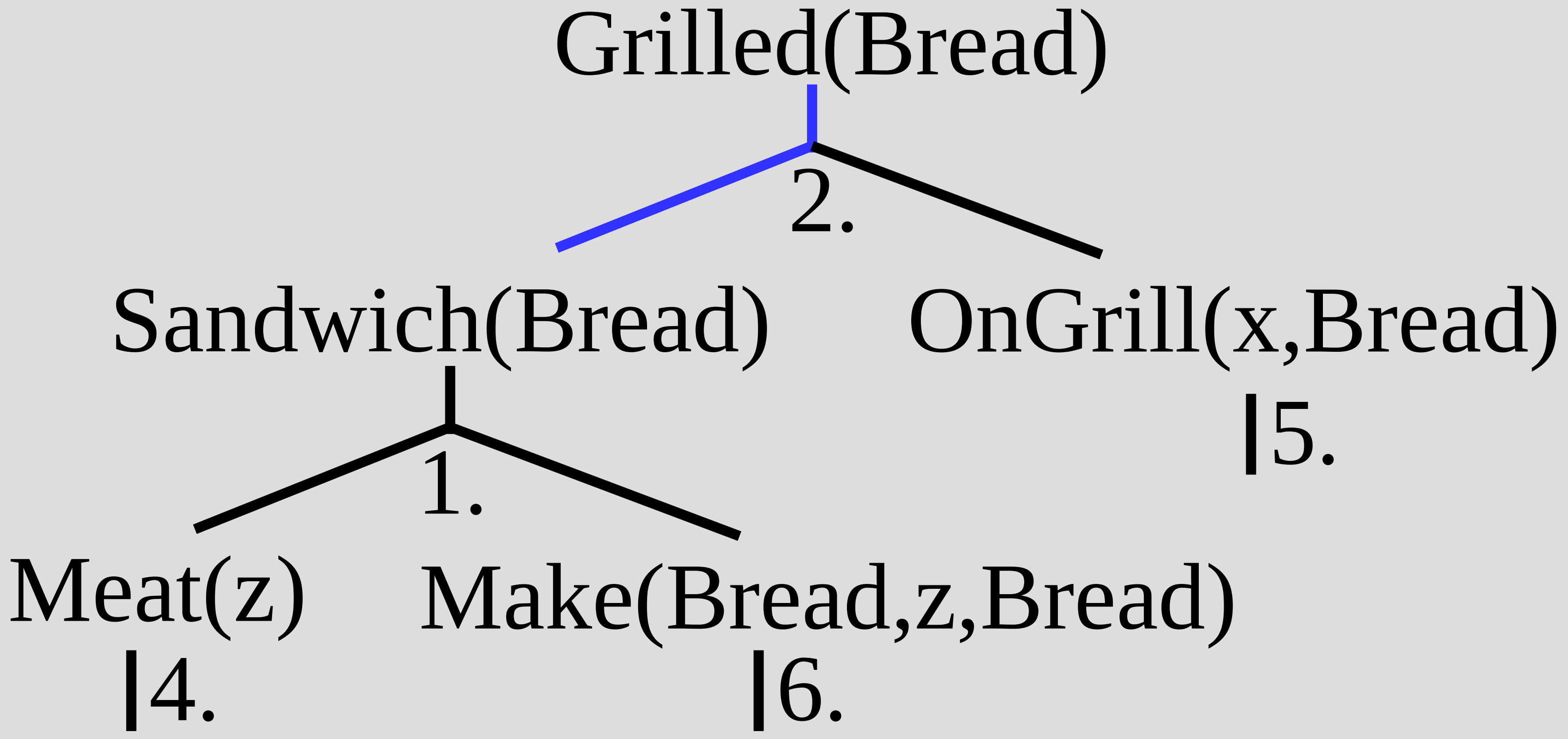
These variables have no correlation, so relabel one to be different

Backward chaining



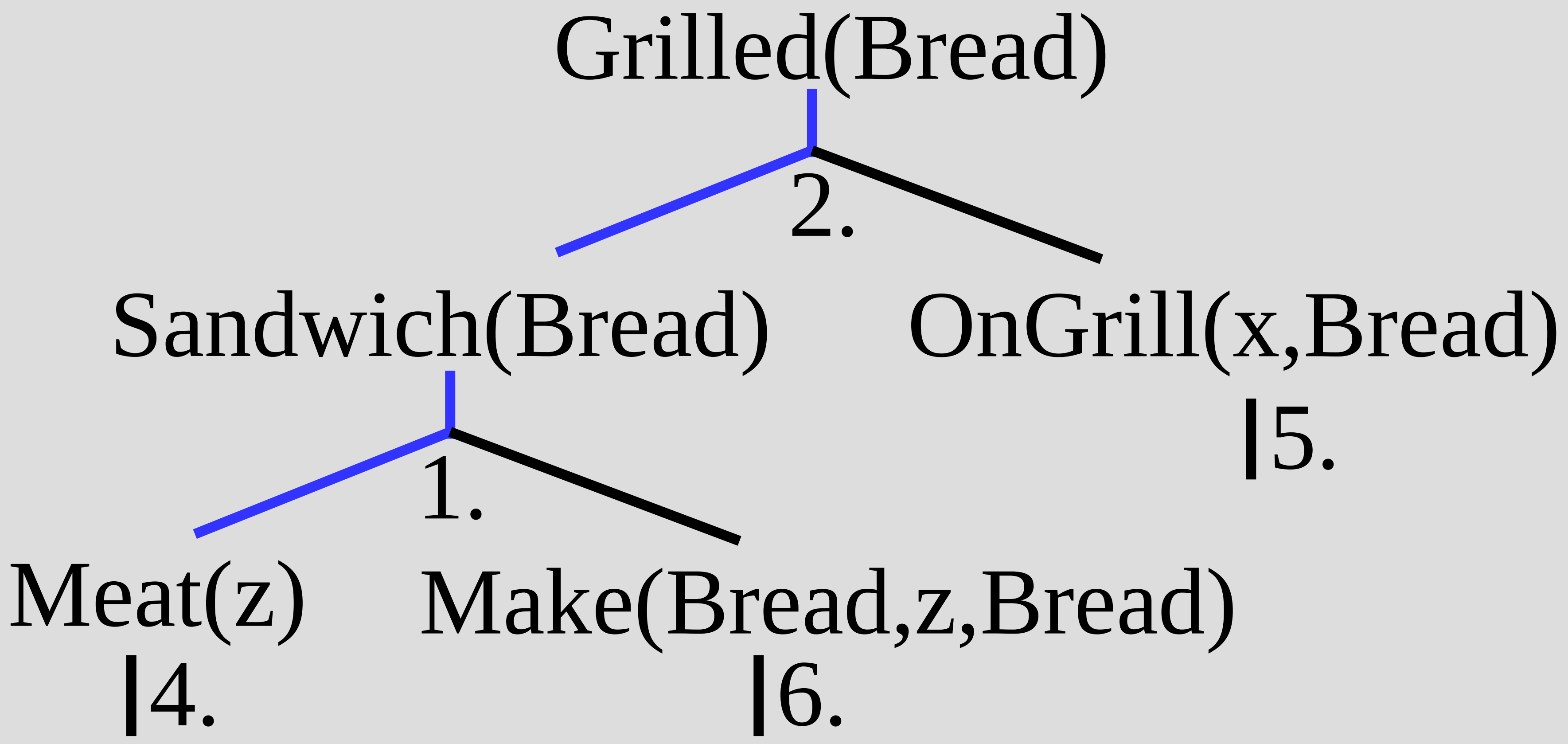
Begin DFS (left branch first)

Backward chaining



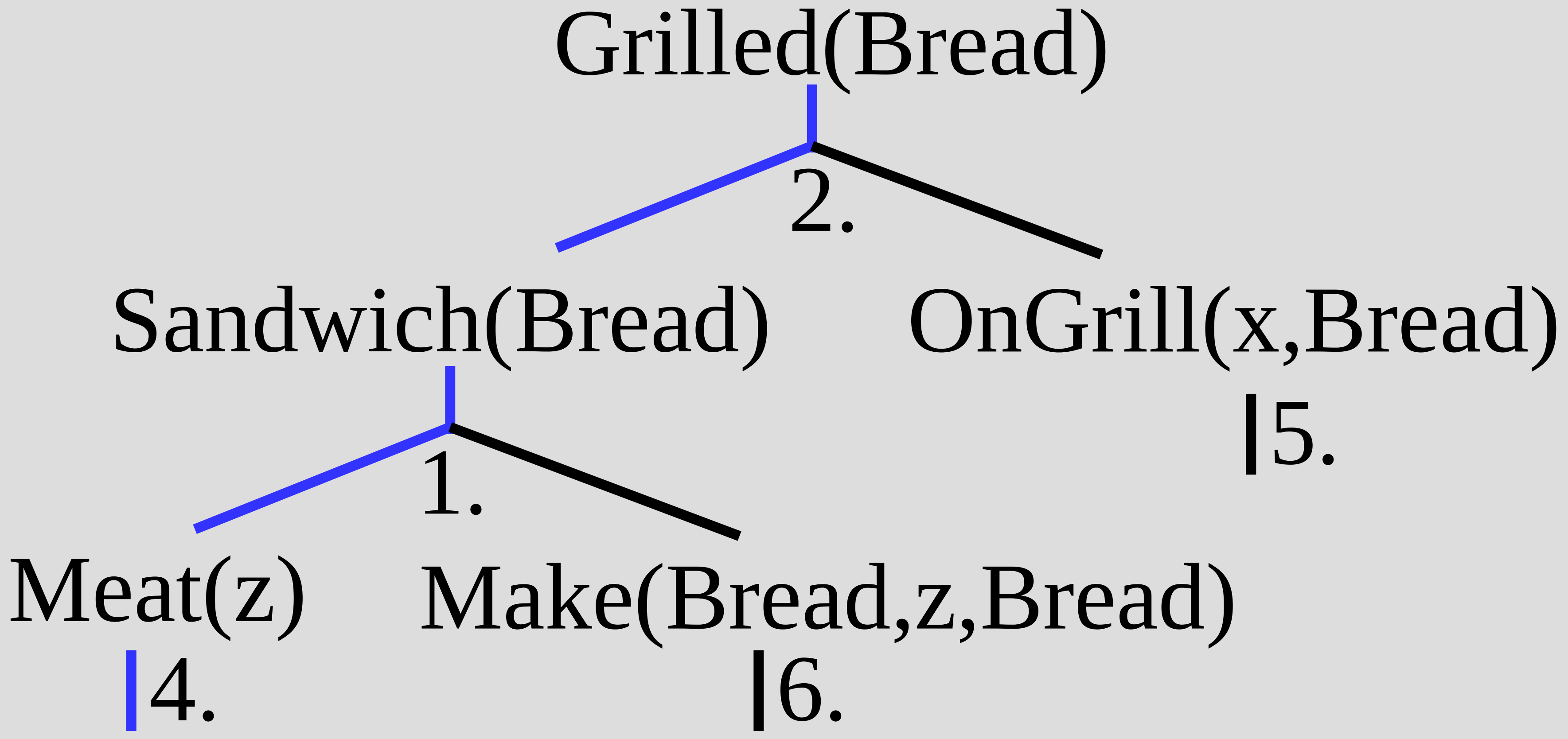
Begin DFS (left branch first)

Backward chaining



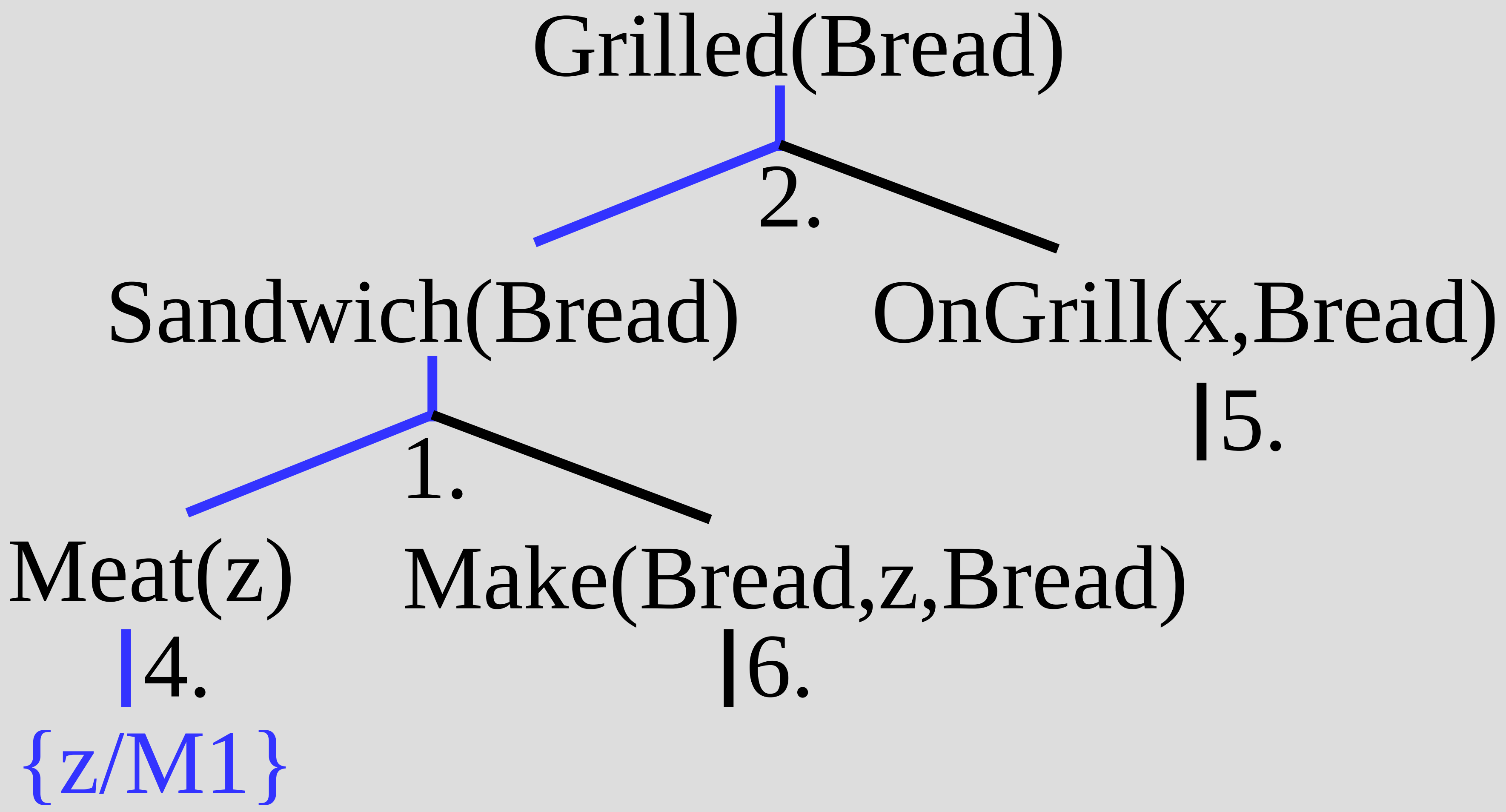
Begin DFS (left branch first)

Backward chaining



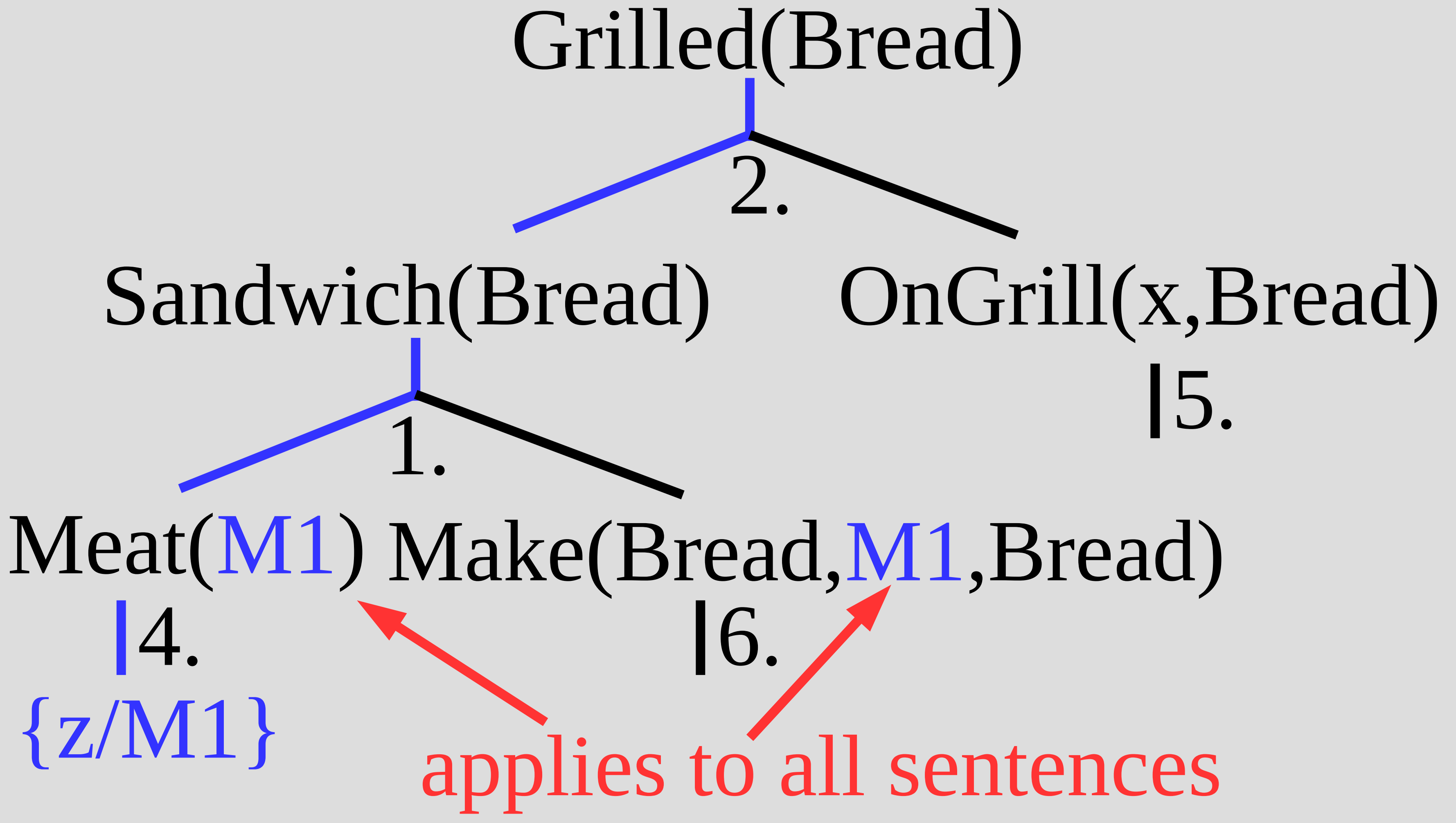
Begin DFS (left branch first)

Backward chaining



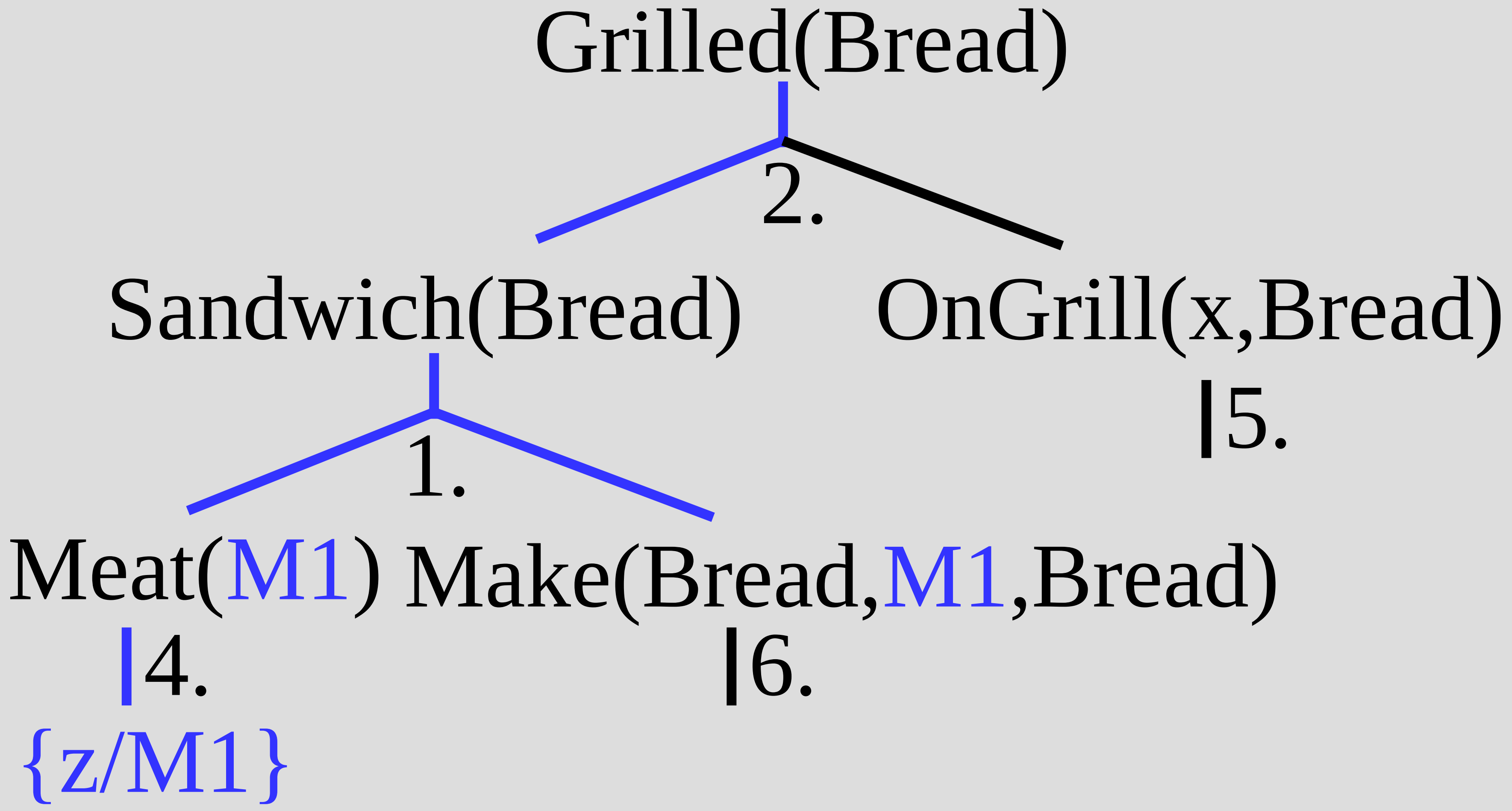
Begin DFS (left branch first)

Backward chaining



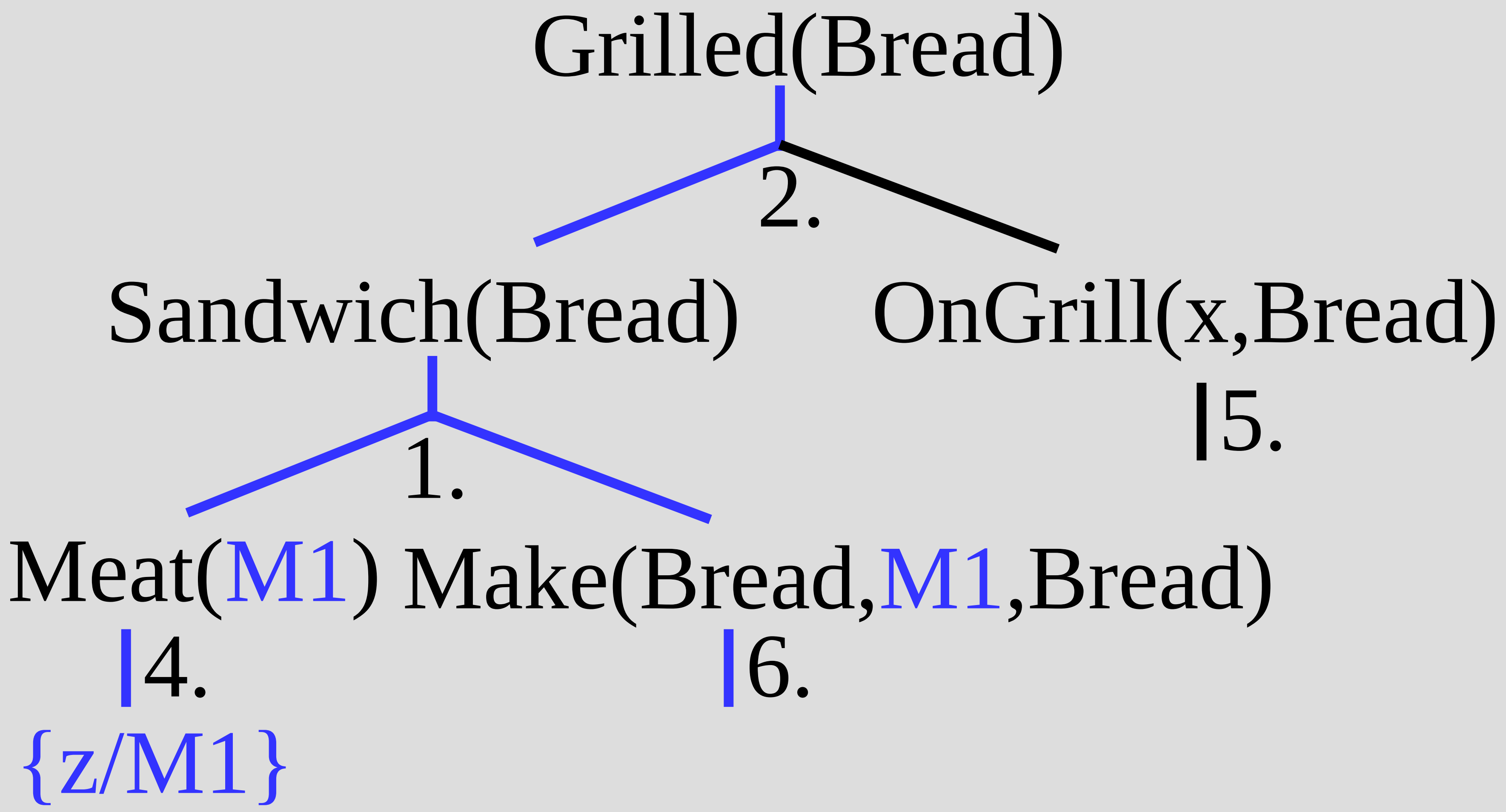
Begin DFS (left branch first)

Backward chaining



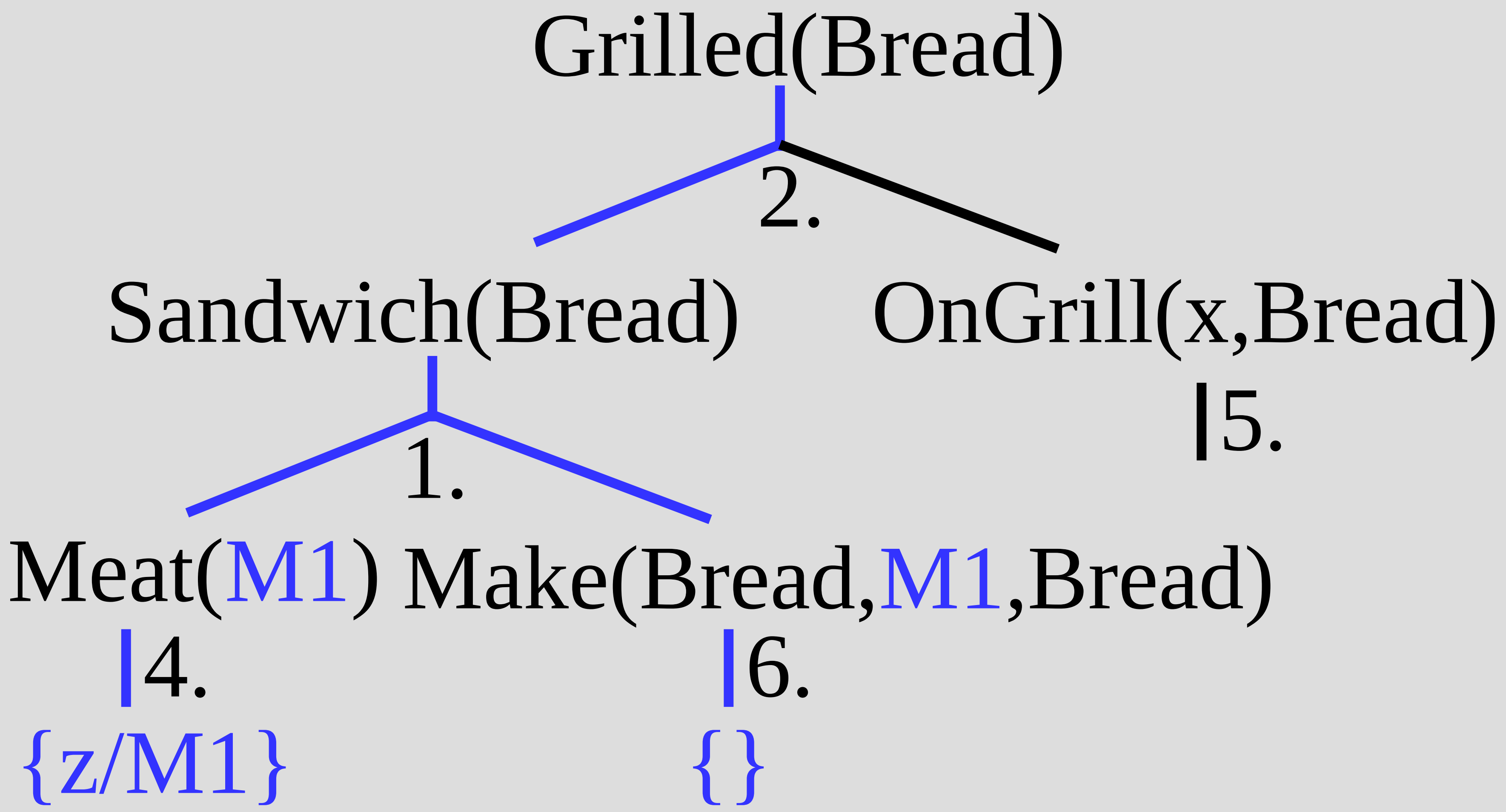
Begin DFS (left branch first)

Backward chaining



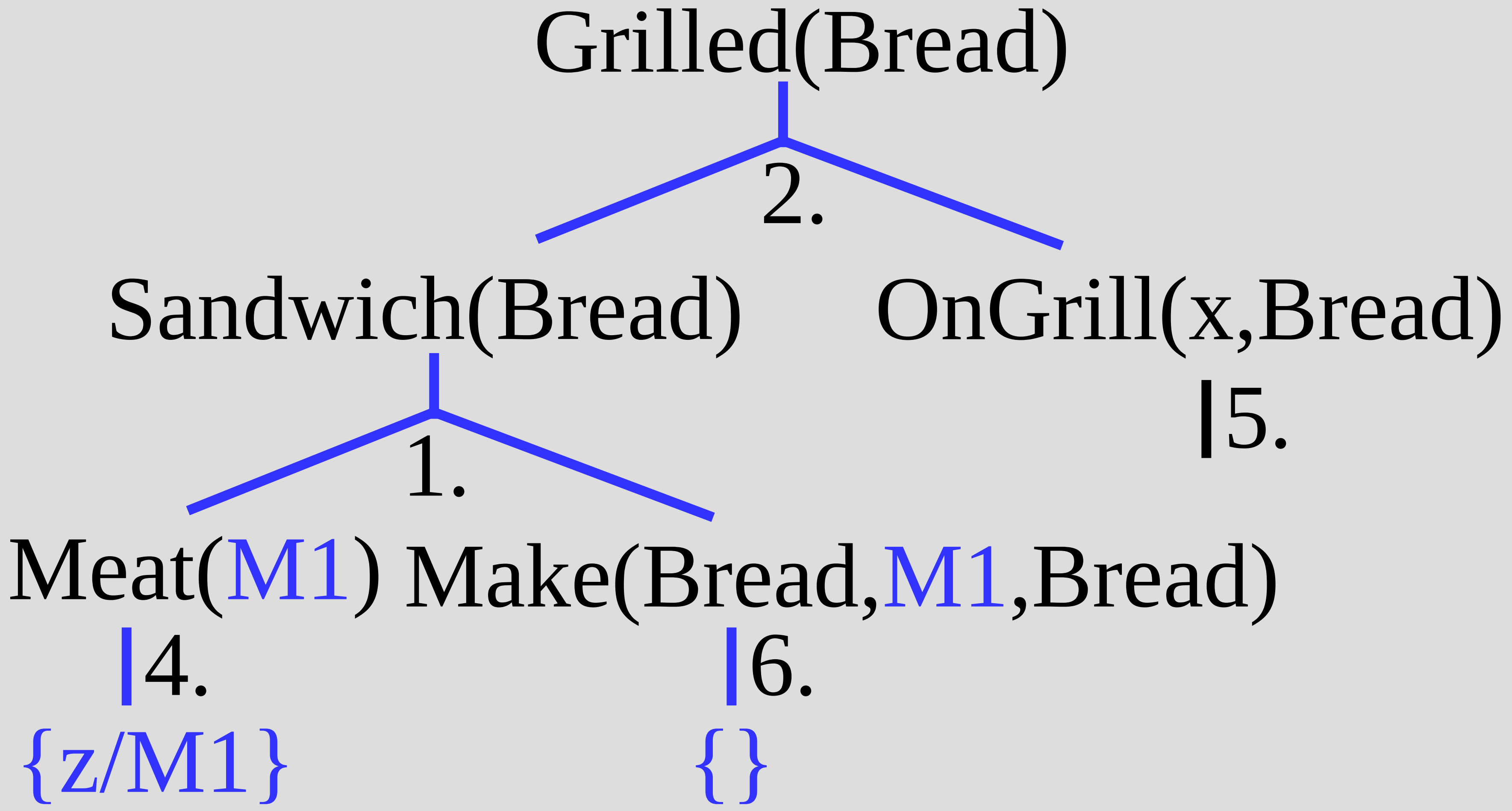
Begin DFS (left branch first)

Backward chaining



Begin DFS (left branch first)

Backward chaining



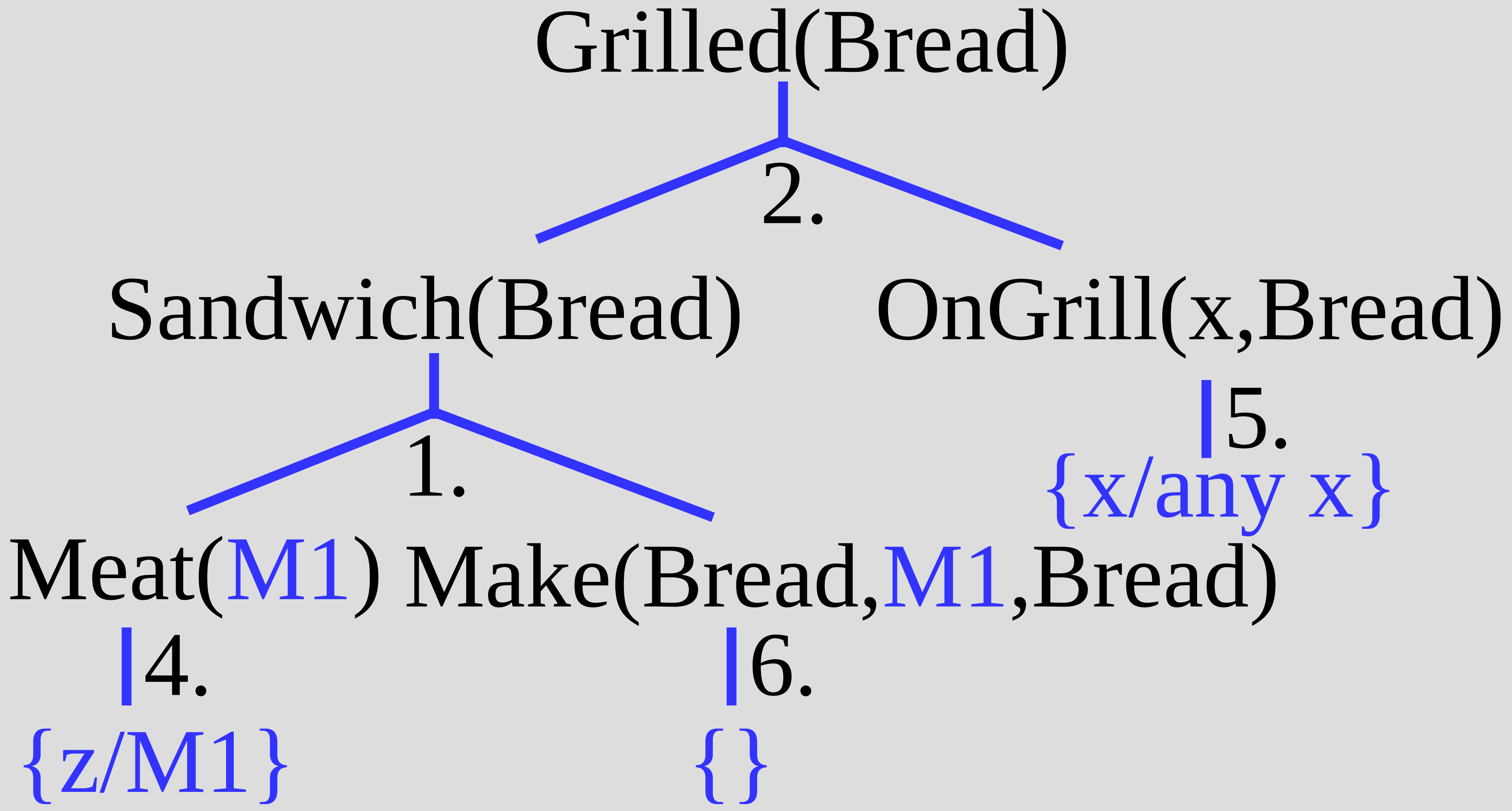
Begin DFS (left branch first)

Backward chaining

$$\begin{tikzcd}
 & & \text{Grilled(Bread)} \arrow[dl, dash, "2."'] \arrow[dr, dash] & \\
 & \text{Sandwich(Bread)} \arrow[dl, dash, "1."'] \arrow[dr, dash] & & \text{OnGrill(x,Bread)} \arrow[d, dash, "5."] \\
\text{Meat(M1)} \arrow[d, dash, "4."] & & \text{Make(Bread,M1,Bread)} \arrow[d, dash, "6."] & {} \\
\{z/M1\} & & \{\} &
\end{tikzcd}$$

Begin DFS (left branch first)

Backward chaining



Begin DFS (left branch first)

Backward chaining

The algorithm to compute this needs to mix between going deeper into the tree (ANDs) and unifying/substituting (ORs)

For this reason, the search is actually two different mini-algorithms that intermingle:

1. FOL-BC-OR (unify)
2. FOL-BC-AND (depth)

Backward chaining

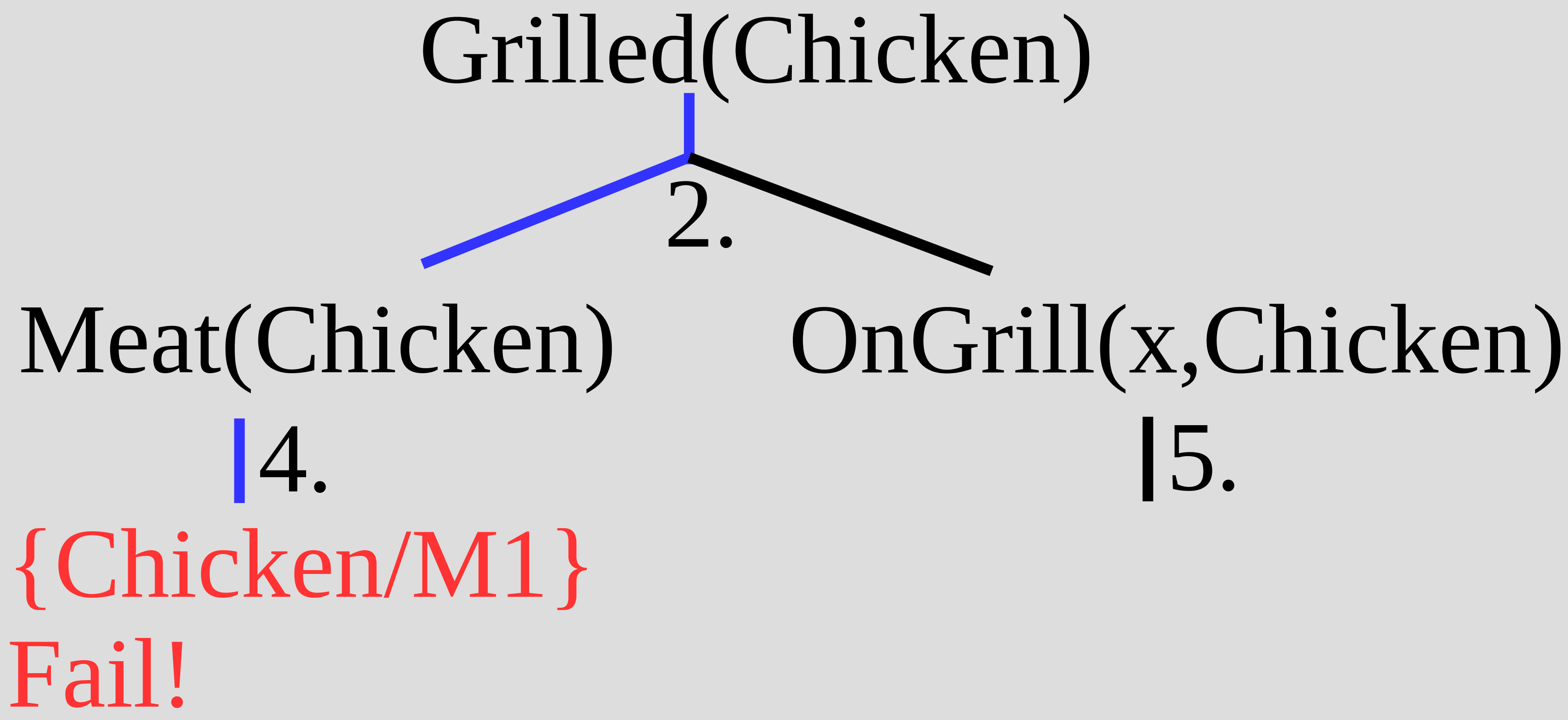
```
FOL-BC-OR(KB, goal, sub)
1.  for each rule (lhs => rhs) with rhs == goal
2.      standardize-variables(lhs, rhs)
3.      for each newSub in FOL-BC-AND(KB, lhs, unify(rhs, goal sub))
4.          yield newSub

FOL-BC-AND(KB, goals sub)
1.  if sub = failure, return
2.  else if length(goals) == 0 then yield sub
3.  else
4.      first,rest ← First(goals), Rest(goals)
5.      for each newSub in FOL-BC-OR(KB, substitute(sub, first), sub)
6.          for each newNewSub in FOL-BC-AND(KB, rest, newSub)
7.              yield newNewSub
```

Backward chaining

Use backward chaining to infer:
Grilled(Chicken)

1. $\forall x\ Meat(x) \land Make(Bread, x, Bread) \Rightarrow Sandwich(Bread)$
2. $\forall x, y\ OnGrill(x, y) \land Sandwich(y) \Rightarrow Grilled(y)$
3. $\forall x, y\ OnGrill(x, y) \land Meat(y) \Rightarrow Grilled(y)$
4. $\exists x\ Meat(x)$
5. $\forall x, y\ OnGrill(x, y)$
6. $\forall x, y, z\ Make(x, y, z)$

Backward chaining



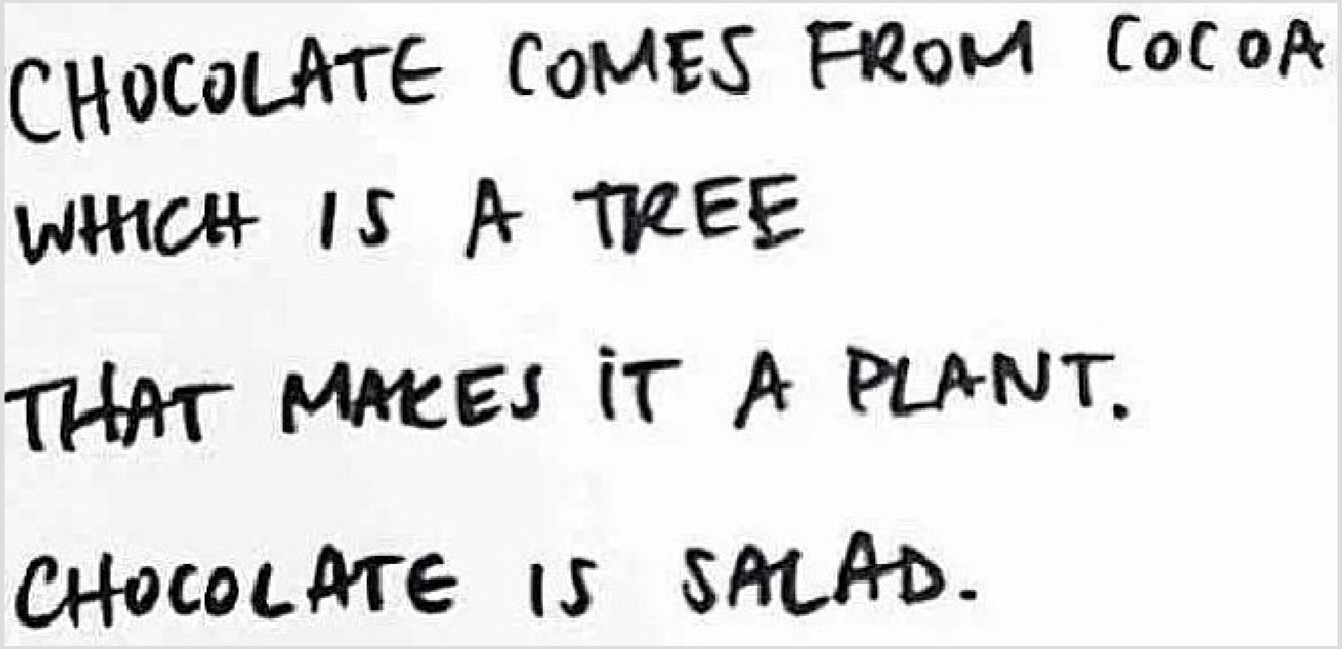
Begin DFS (left branch first)

Backward chaining

Similar to normal DFS, this backward chaining can get stuck in infinite loops (in the case of functions)

However, in general it can be much faster as it can be fairly easily parallelized (the different branches of the tree)

Resolution in FO logic (Ch. 9)


CHOCOLATE COMES FROM COCOA
WHICH IS A TREE
THAT MAKES IT A PLANT.
CHOCOLATE IS SALAD.

Review: CNF form

Conjunctive normal form is a number of clauses stuck together with ANDs

Each clause can only contain ORs, and logical negation must appears right next to literals

For example: CNF with 3 clauses

$$\underline{(A)} \land \underline{(A \lor \neg B)} \land \underline{(\neg A \lor \neg B \lor C \lor D)}$$

clauses

First-order logic resolution

To do first-order logic resolution we again need to get all the sentences to CNF

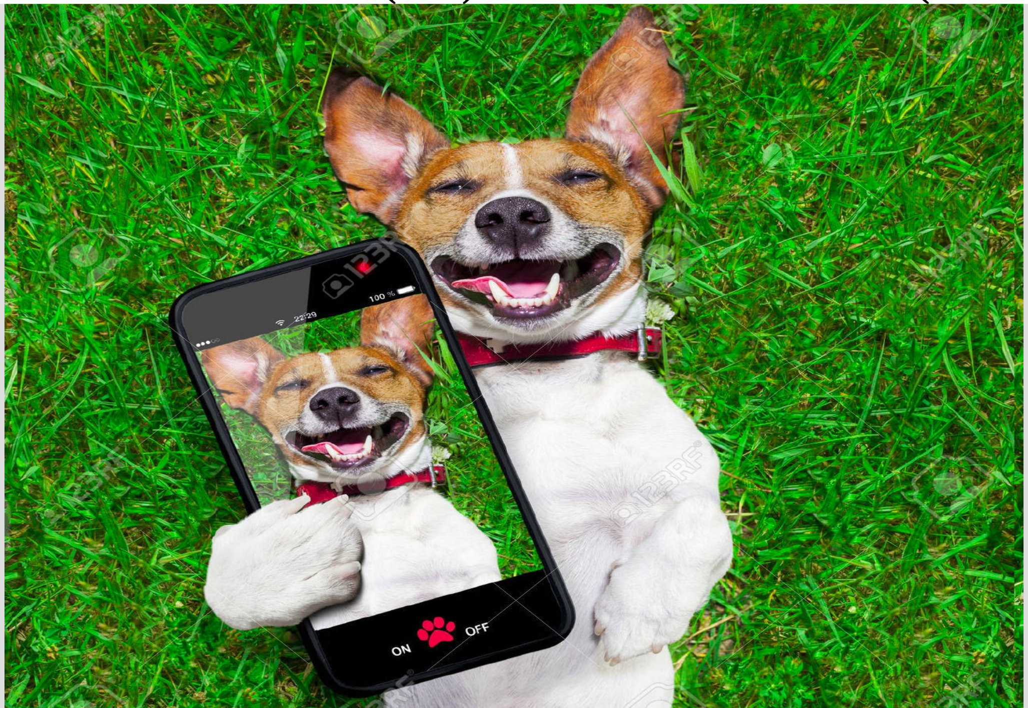
This requires a few more steps for FOL (red):

1. Use logical equivalence to remove implies
2. Move logical negation next to relations
3. Standardize variables
4. Generalize existential quantifiers
5. Drop universal quantifiers
6. Distribute ORs over ANDs

First-order logic resolution

"All dogs that are able to make everyone laugh are owned by someone"

$$\forall x\ [Dog(x) \land \forall y\ Person(y) \land Laugh(y,x)] \Rightarrow [\exists y\ Person(y) \land Owns(y,x)]$$



First-order logic resolution

$$\forall x\ [Dog(x) \land \forall y\ Person(y) \land Laugh(y,x)]$$
$$\Rightarrow [\exists y\ Person(y) \land Owns(y,x)]$$

I have avoided putting quantifiers anywhere except the left for simplicity (as you will see)

There is always a equivalent form with all quantifiers to the left of the main sentence

But the above sentence is logically valid

1. convert implies

As CNF only has ORs and ANDs, we use this:

$$A \Rightarrow B \equiv \neg A \vee B$$

If there is a $\iff$, we use the following first:

$$A \iff B \equiv A \Rightarrow B \wedge B \Rightarrow A$$

First-order logic only allows these logical ops: $\neg, \vee, \wedge, \Rightarrow, \iff$

So we will have reduced everything to just negation, ANDs and ORs

1. convert implies

$$\forall x \; [Dog(x) \wedge \forall y \; Person(y) \wedge Laugh(y,x)] \\ \Rightarrow [\exists y \; Person(y) \wedge Owns(y,x)]$$

... converts to...

$$\forall x \; \neg[Dog(x) \wedge \forall y \; Person(y) \wedge Laugh(y,x)] \\ \vee[\exists y \; Person(y) \wedge Owns(y,x)]$$

This is now the statement:
“Dogs are either not thought as funny by everyone or owned by someone”

2. move negation to right

Putting negation next to relationships requires two things:

1. De Morgan's laws:

$$\neg(A \vee B) \equiv (\neg A \wedge \neg B)$$
$$\neg(A \wedge B) \equiv (\neg A \vee \neg B)$$

2. Quantifier negation:

$$\neg(\forall x \ A(x)) \equiv (\exists x \ \neg A(x))$$
$$\neg(\exists x \ A(x)) \equiv (\forall x \ \neg A(x))$$

2. move negation to right

$$\forall x \neg[Dog(x) \wedge \forall y\ Person(y) \wedge Laugh(y,x)]$$
$$\vee[\exists y\ Person(y) \wedge Owns(y,x)]$$

... converts to...

$$\forall x\ [\neg Dog(x) \vee \exists y\ \neg Person(y) \vee \neg Laugh(y,x)]$$
$$\vee[\exists y\ Person(y) \wedge Owns(y,x)]$$

This is now the statement: “All things are either (not a dog or not funny to a human or funny to a non-human) or owned by someone”

3. standardize variables

It is possible to reuse the same variable in multiple parts of a sentence, such as y in:

$$\forall x\ (\exists y\ A(x,y)) \Rightarrow (\exists y\ B(x,y))$$

You can just rename a variable to make it clear that there is no conflict (having quantifiers on the left ensures there is no confusion)

rename this y to z

$$\forall x\ (\exists y\ A(x,y)) \Rightarrow (\exists \overset{\swarrow}{y}\ B(x,y))$$
$$\equiv \forall x \exists y, z\ A(x,y) \Rightarrow B(x,z)$$

3. standardize variables

$$\forall x\ [\neg Dog(x) \vee \exists y\ \neg Person(y) \vee \neg Laugh(y, x)]$$
$$\vee[\exists y\ Person(y) \wedge Owns(y, x)]$$

... converts to...

$$\forall x, \exists y, z\ [\neg Dog(x) \vee \neg Person(y) \vee \neg Laugh(y, x)]$$
$$\vee[Person(z) \wedge Owns(z, x)]$$

The meaning is still the same as last time, but might be easier to understand in half-English: “Every x is either (not a dog, not funny to y or y is not a person) or (person z owns x)”

4. generalize existential

We have talked before about how to make a new object for an existential quantifier:

$$\text{Objects} = \{\text{a, b, c}\} \quad \longrightarrow \quad \text{Objects} = \{\text{a, b, c, A1}\}$$
$$\exists x\ A(x) \qquad\qquad \text{A(A1)}$$

However, the situation is more difficult for existential inside universal quantifier:

$$\text{Objects} = \{\text{a, b, c}\} \quad \xrightarrow[??]{} \quad \text{Objects} = \{\text{a, b, c, A1}\}$$
$$\forall x \exists y\ A(x, y) \qquad\qquad \forall x\ A(x, A1)$$

Does this work?

4. generalize existential

This does not work...

$$\text{Objects} = \{\text{a, b, c, A1}\} \quad \forall x\ A(x, A1) \rightarrow \text{Objects} = \{\text{a, b, c}\} \quad \exists y \forall x\ A(x, y)$$

This is saying there is a single object (A1), which is true for all x

To properly represent existential on the inside, we need to use a function of x to represent y:

$$\text{Objects} = \{\text{a, b, c}\} \quad \forall x \exists y\ A(x, y) \rightarrow \text{Objects} = \{\text{a, b, c}\} \quad \forall x\ A(x, F(x))$$

4. generalize existential

Function review:
Unary relations = Person(x) (is a relation)
Function = child(x) (is an object)
(functions can also have more than one input)

$$\text{Objects} = \{\text{a, b, c}\} \qquad \text{Objects} = \{\text{a, b, c}\}$$
$$\forall x \exists y\ A(x, y) \rightarrow \forall x\ A(x, F(x))$$

Here the function F(x) is the y for which A(x,y) is true for any given x
(this is called Skolemization)

4. generalize existential

$$\forall x, \exists y, z \ [\neg Dog(x) \vee \neg Person(y) \vee \neg Laugh(y,x)]$$
$$\vee [Person(z) \wedge Owns(z,x)]$$

... converts to...

$$\forall x \ [\neg Dog(x) \vee \neg Person(Y(x)) \vee \neg Laugh(Y(x),x)]$$
$$\vee [Person(Z(x)) \wedge Owns(Z(x),x)]$$

... I give up translating

If there were multiple universal quantifiers, all the variables would be in the function:

$$\forall x, y \exists a \forall z \ A(x,y,z,a) \equiv \forall x, y, z \ A(x,y,z,F(x,y))$$

5. drop universal quantifiers

As we got rid of existential, there is no confusion about the quantifiers...

So we just simply drop the “for all”s:

$$\forall x\ [\neg Dog(x) \vee \neg Person(Y(x)) \vee \neg Laugh(Y(x), x)] \\ \vee [Person(Z(x)) \wedge Owns(Z(x), x)]$$

... converts to...

$$[\neg Dog(x) \vee \neg Person(Y(x)) \vee \neg Laugh(Y(x), x)] \\ \vee [Person(Z(x)) \wedge Owns(Z(x), x)]$$

6. distribute AND/OR

To get in CNF form, we need all clauses to only contain ORs, and be separated by ANDs:

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$

(basic logic rules of equivalence)

A	B	C	B^C	AV(B^C)	AVB	AVC	(AVB)^(AVC)
T	T	T	T	T	T	T	T
T	T	F	F	T	T	T	T
T	F	T	F	T	T	T	T
T	F	F	F	T	T	T	T
F	T	T	T	T	T	T	T
F	T	F	F	F	T	F	F
F	F	T	F	F	F	T	F
F	F	F	F	F	F	F	F

6. distribute AND/OR

$$A = [\neg Dog(x) \vee \neg Person(Y(x)) \vee \neg Laugh(Y(x), x)]$$

$$B = Person(Z(x))$$

$$C = Owns(Z(x), x)$$

- Substitute into:

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$

$$[\neg Dog(x) \vee \neg Person(Y(x)) \vee \neg Laugh(Y(x), x)]$$
$$\vee [Person(Z(x)) \wedge Owns(Z(x), x)]$$

- ... converts to...

$$\left(\neg Dog(x) \vee \neg Person(Y(x)) \vee \neg Laugh(Y(x), x) \vee Person(Z(x))\right)$$
$$\wedge \left(\neg Dog(x) \vee \neg Person(Y(x)) \vee \neg Laugh(Y(x), x) \vee Owns(Z(x), x)\right)$$

Resolution in FO logic

Once you have the first-order logic in CNF-ish form, resolution is almost the same

The only difference is that you must unify/ substitute any variables that you merge

For example:

$$\big(A(x, y, Z(x)) \vee \neg B(y)\big) \wedge \big(C(x) \vee B(Y(x))\big)$$

... unify/substitute {y/Y(x)}

$$\big(A(x, Y(x), Z(x)) \vee C(x)\big)$$