Logic Programming in Prolog (PLP 11)

Terms, Resolution, Unification, Search, Backtracking

Carlos Varela

Rensselaer Polytechnic Institute

November 10, 2026

Prolog Terms

- Constants

```
rpi
troy
```

- Variables

```
University
City
```

- Predicates

```
located_at(rpi,troy)
pair(a, pair(b,c))
```

Can be nested.

Resolution

- To derive new statements, Robinson's resolution principle says that if two Horn clauses:

$$H_1 \Leftarrow B_{11}, B_{12}, \ldots, B_{1m}$$
$$H_2 \Leftarrow B_{21}, B_{22}, \ldots, B_{2n}$$

are such that **H_1** matches **B_{2i}**, then we can replace **B_{2i}** with **$B_{11}, B_{12}, \ldots, B_{1m}$** :

$$H_2 \Leftarrow B_{21}, B_{22}, \ldots, B_{2(i-1)}, \underline{B_{11}, B_{12}, \ldots, B_{1m}}, B_{2(i+1)} \ldots, B_{2n}$$

- For example:

$$\begin{array}{l} C \Leftarrow A,B \\ \underline{E \Leftarrow C,D} \\ E \Leftarrow A,B,D \end{array}$$

Resolution Example

```
father(X,Y) :- parent(X,Y), male(X).
grandfather(X,Y) :- father(X,Z), parent(Z,Y).
```

```
grandfather(X,Y) :-
          parent(X,Z), male(X), parent(Z,Y).
```

`:-` is Prolog's notation (syntax) for $\Leftarrow$.

Unification

- During *resolution*, free variables acquire values through *unification* with expressions in matching terms.
- For example:

```
male(carlos).
parent(carlos, tatiana).
parent(carlos, catalina).
father(X,Y) :- parent(X,Y), male(X).
```

```
father(carlos, tatiana).
father(carlos, catalina).
```

Unification Process

- A constant unifies only with itself or a variable.
- Two predicates unify if and only if they have
  - the same *functor*,
  - the same number of *arguments*, and
  - the corresponding arguments *unify*.
- A variable unifies with anything.
  - If the other thing has a *value*, then the variable is *instantiated*.
  - If it is an *uninstantiated variable*, then the two variables are *associated*.

Backtracking

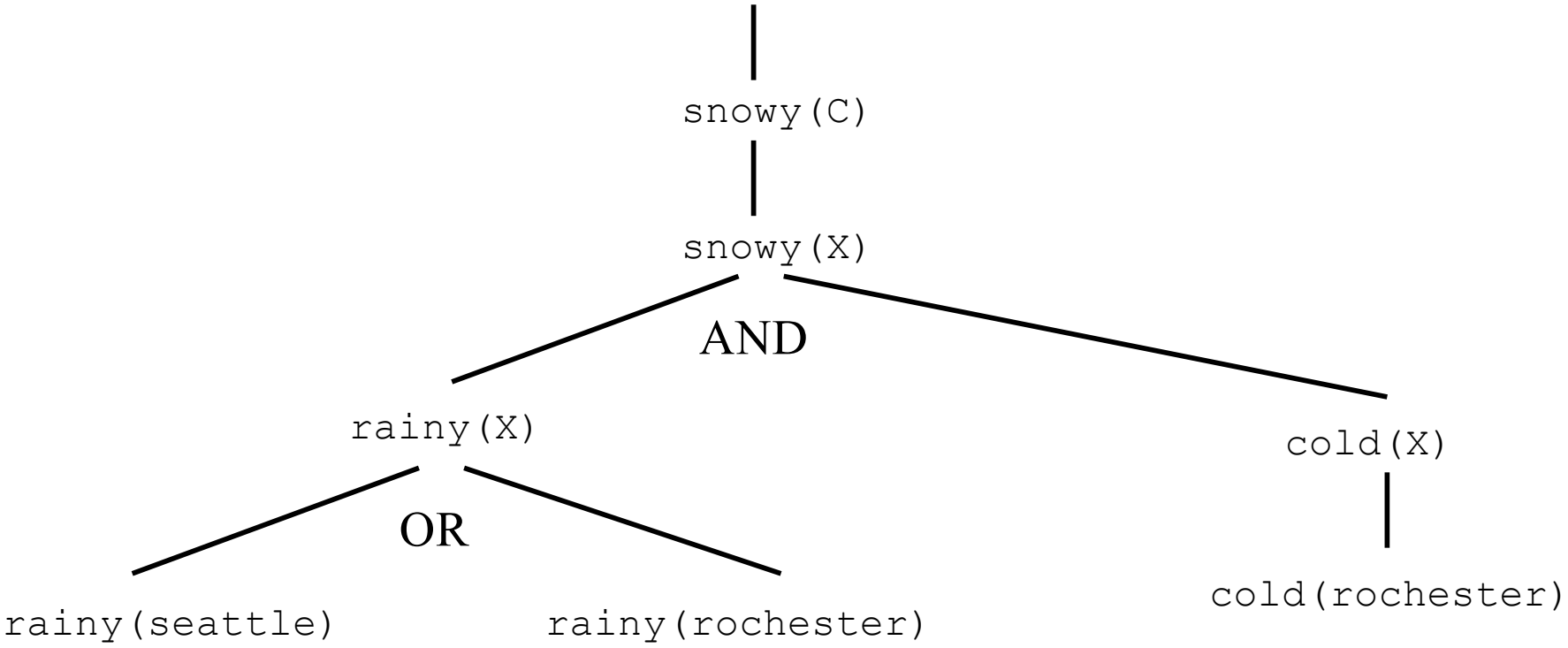
- *Forward chaining* goes from axioms forward into goals.
- *Backward chaining* starts from goals and works backwards to prove them with existing axioms.

Backtracking example

```
rainy(seattle).
rainy(rochester).
cold(rochester).
snowy(X) :- rainy(X), cold(X).
```

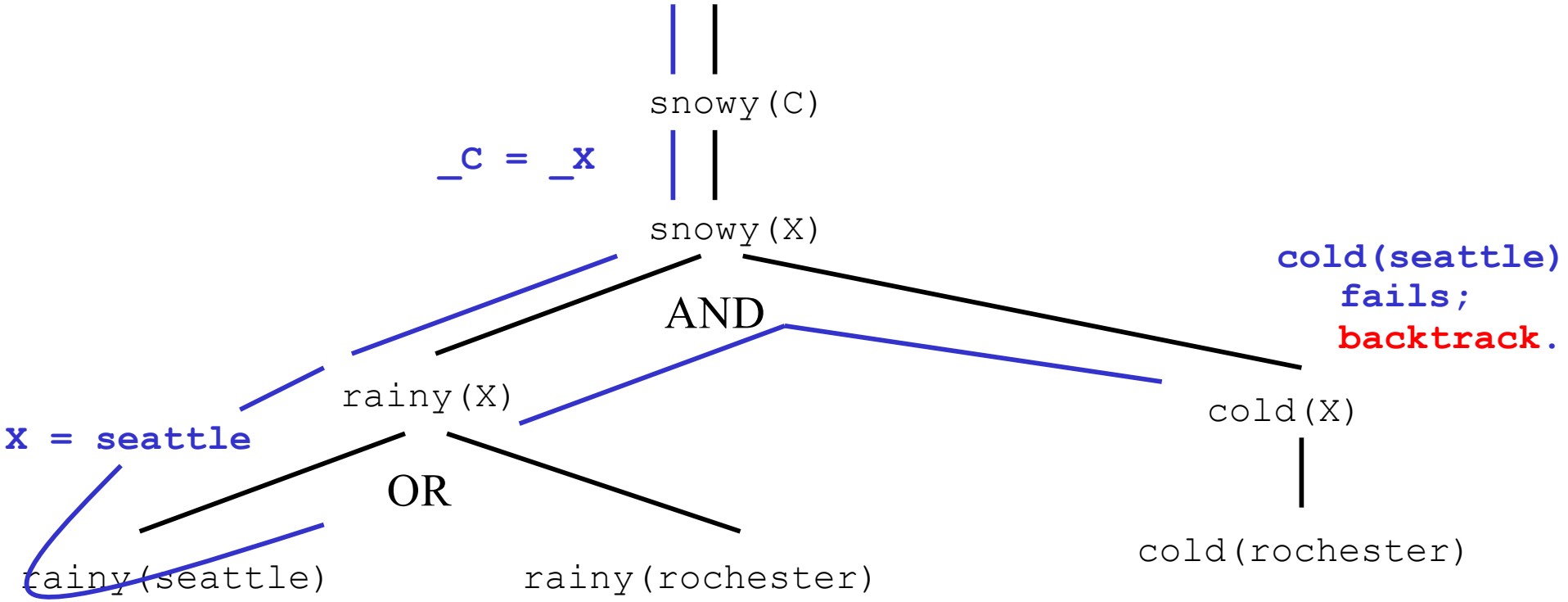
Backtracking example

```
rainy(seattle).
rainy(rochester).
cold(rochester).
snowy(X) :- rainy(X), cold(X).
```



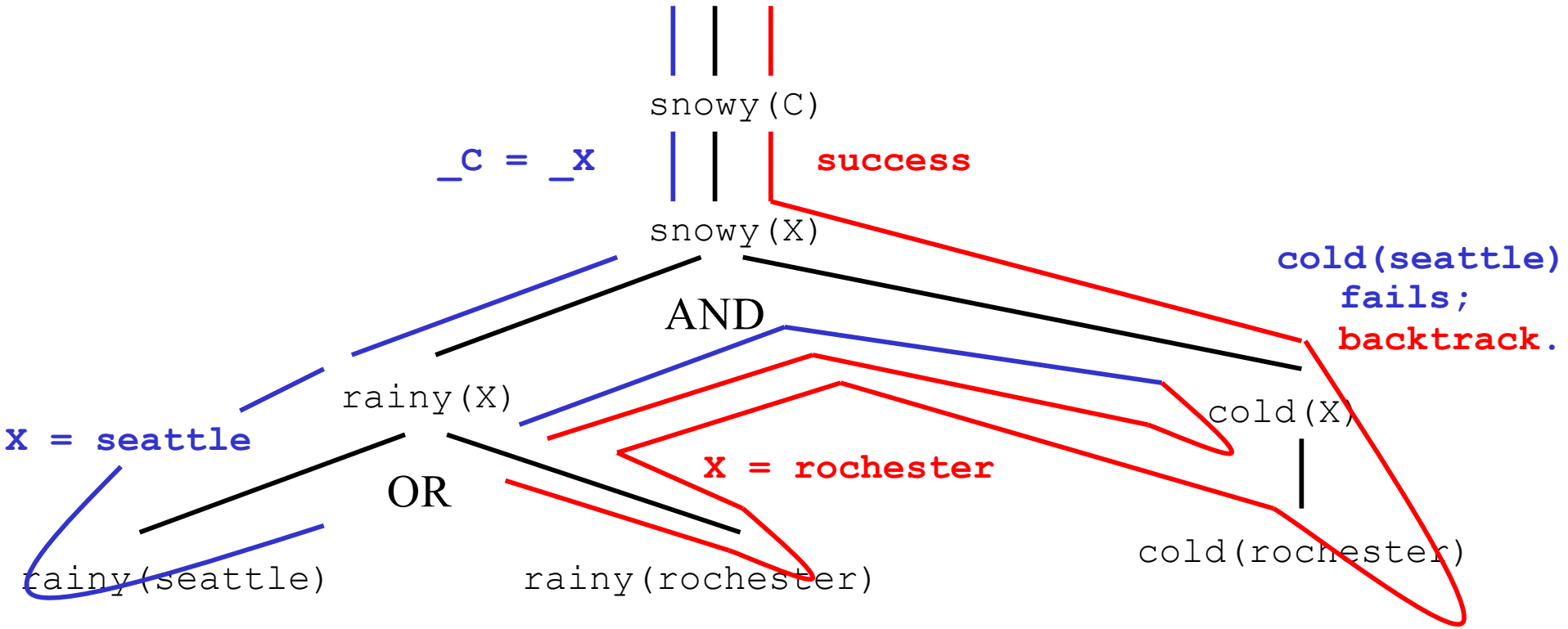
Backtracking example

```
rainy(seattle).
rainy(rochester).
cold(rochester).
snowy(X) :- rainy(X), cold(X).
```



Backtracking example

```
rainy(seattle).
rainy(rochester).
cold(rochester).
snowy(X) :- rainy(X), cold(X).
```



Exercises

71. Download SWI Prolog and install it in your laptop.
72. Execute the "`snowy(City)`" example. Use "tracing" to follow backtracking step by step.
73. Create a knowledge base with facts about your family members using predicates and constants. Create rules using variables to define the following relationships: `brother`, `sister`, `uncle`, `aunt`, `nephew`, `niece`, `grandfather`, `grandmother`, etc. Query your Prolog program for family relationships.