

# Logic Programming (PLP 11, CTM 9.1)

Terms, Resolution, Unification, Search, Backtracking  
(Prolog)

Relational Computation Model (Oz)

Carlos Varela

Rensselaer Polytechnic Institute

November 12, 2021

# 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:

$$H_1 \leftarrow B_{11}, B_{12}, \dots, B_{1m}$$

$$H_2 \leftarrow B_{21}, B_{22}, \dots, B_{2n}$$

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

$$H_2 \leftarrow B_{21}, B_{22}, \dots, B_{2(i-1)}, \underbrace{B_{11}, B_{12}, \dots, B_{1m}}, B_{2(i+1)}, \dots, B_{2n}$$

- For example:

$$C \leftarrow A, B$$

$$E \leftarrow C, D$$

$$\hline E \leftarrow A, B, D$$

# 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.
- 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.

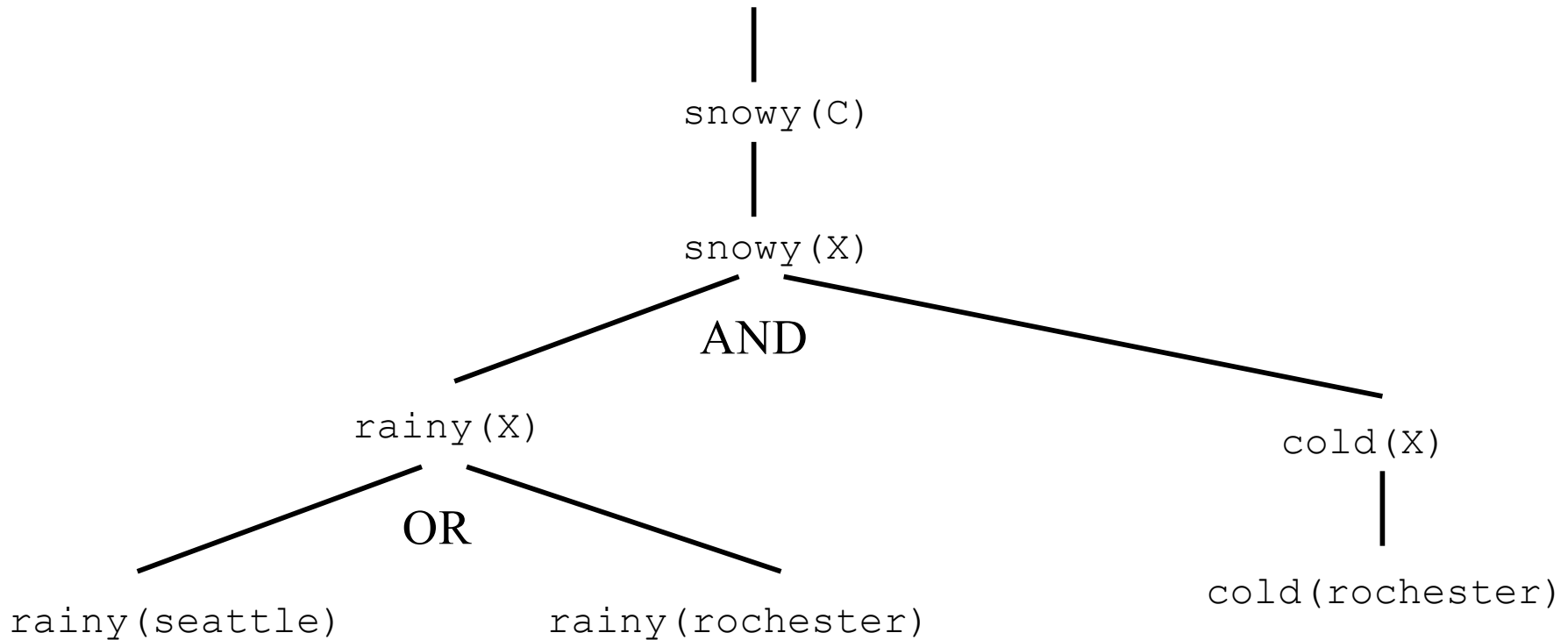
# Backtracking example

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



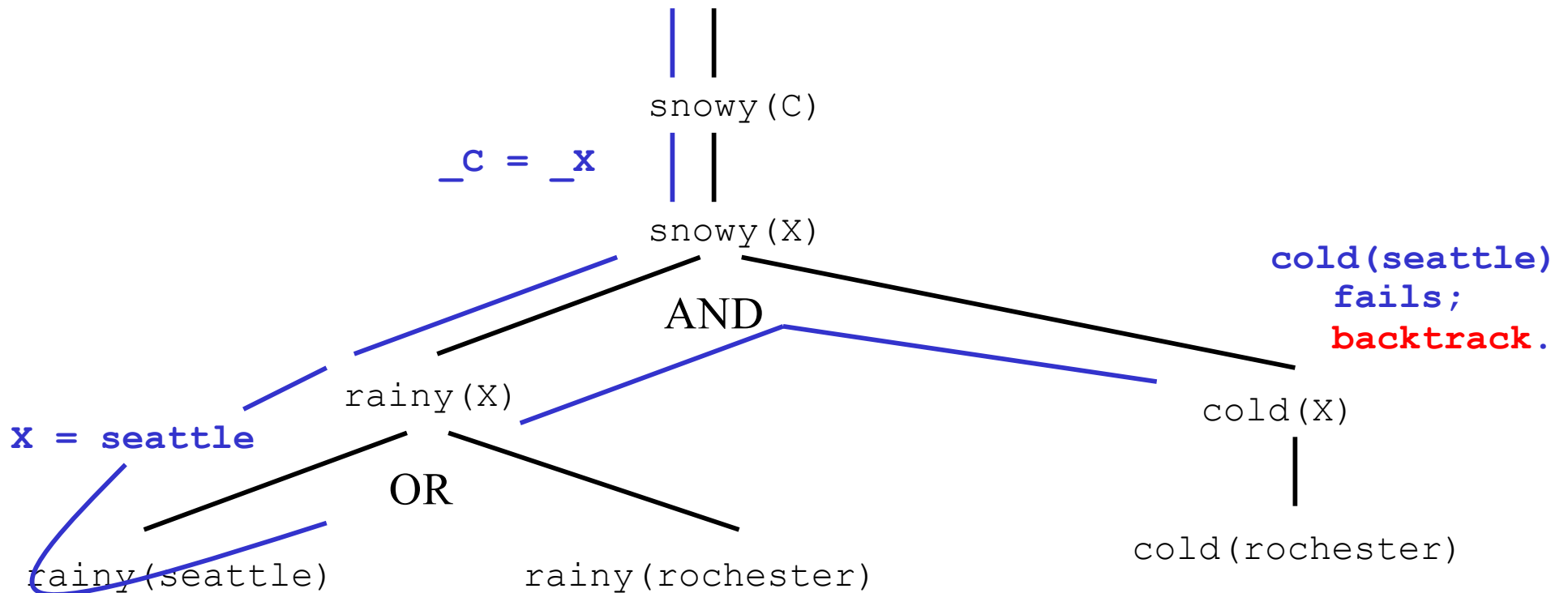
# Backtracking example

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



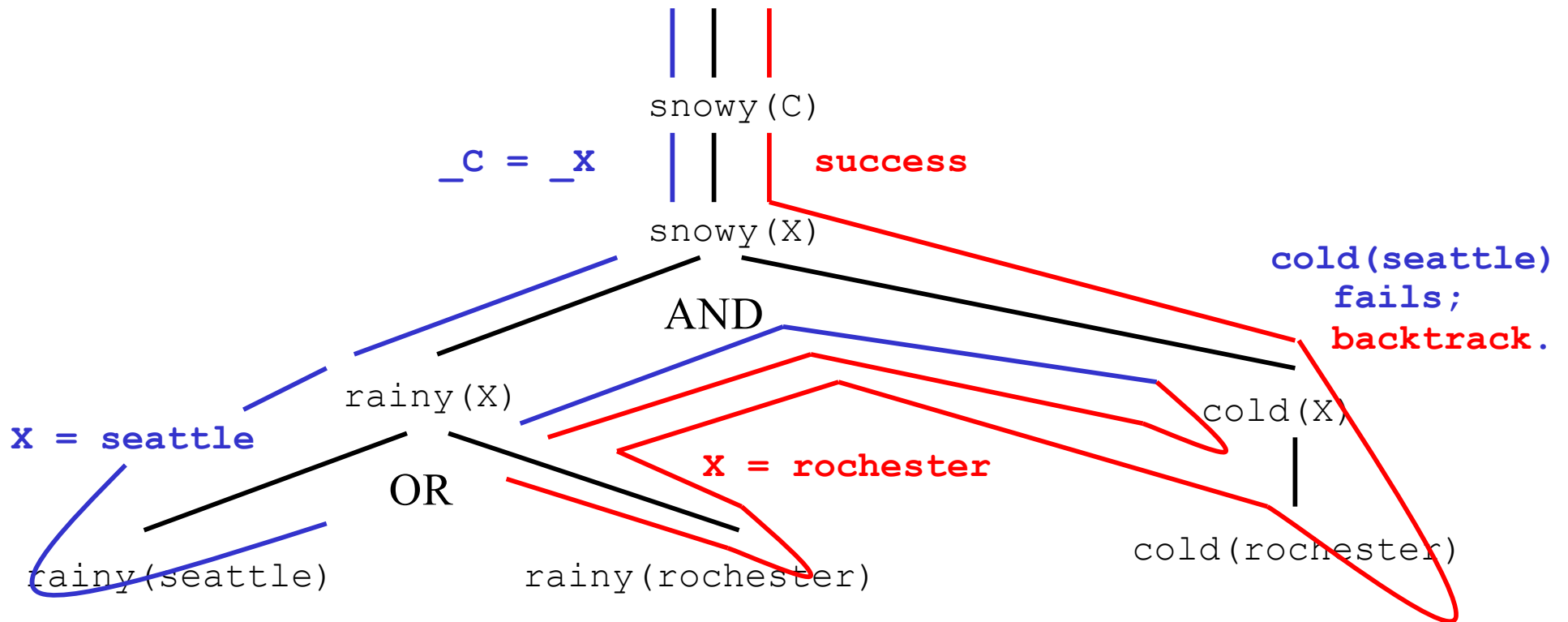
# Backtracking example

```
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).
```



# Relational computation model (Oz)

The following defines the syntax of a statement,  $\langle s \rangle$  denotes a statement

$\langle s \rangle ::=$	<b>skip</b>	<i>empty statement</i>
	$\langle x \rangle = \langle y \rangle$	<i>variable-variable binding</i>
	$\langle x \rangle = \langle v \rangle$	<i>variable-value binding</i>
	$\langle s_1 \rangle \langle s_2 \rangle$	<i>sequential composition</i>
	<b>local</b> $\langle x \rangle$ <b>in</b> $\langle s_1 \rangle$ <b>end</b>	<i>declaration</i>
	<b>proc</b> $\{ \langle x \rangle \langle y_1 \rangle \dots \langle y_n \rangle \}$ $\langle s_1 \rangle$ <b>end</b>	<i>procedure</i>
<i>introduction</i>		
	<b>if</b> $\langle x \rangle$ <b>then</b> $\langle s_1 \rangle$ <b>else</b> $\langle s_2 \rangle$ <b>end</b>	<i>conditional</i>
	$\{ \langle x \rangle \langle y_1 \rangle \dots \langle y_n \rangle \}$	<i>procedure application</i>
	<b>case</b> $\langle x \rangle$ <b>of</b> $\langle \text{pattern} \rangle$ <b>then</b> $\langle s_1 \rangle$ <b>else</b> $\langle s_2 \rangle$ <b>end</b>	<i>pattern matching</i>
	<b>choice</b> $\langle s_1 \rangle$ <b>[]</b> $\dots$ <b>[]</b> $\langle s_n \rangle$ <b>end</b>	<b>choice</b>
	<b>fail</b>	<b>failure</b>

# Relational Computation Model

- Declarative model (purely functional) is extended with *relations*.
- The **choice** statement groups a set of alternatives.
  - Execution of choice statement chooses one alternative.
  - Semantics is to rollback and try other alternatives if a failure is subsequently encountered.
- The **fail** statement indicates that the current alternative is wrong.
  - A **fail** is implicit upon trying to bind incompatible values, e.g.,  $3=4$ . This is in contrast to raising an exception (as in the declarative model).

# Search tree and procedure

- The search tree is produced by creating a new branch at each *choice point*.
- When **fail** is executed, execution « backs up » or backtracks to the most recent **choice** statement, which picks the next alternative (left to right).
- Each path in the tree can correspond to no solution (« fail »), or to a solution (« succeed »).
- A search procedure returns a lazy list of all solutions, ordered according to a depth-first search strategy.

# Rainy/Snowy Example

```
fun {Rainy}
  choice
    seattle [] rochester
  end
end
```

```
fun {Cold}
  rochester
end
```

```
proc {Snowy X}
  {Rainy X}
  {Cold X}
end
```

```
{Browse
  {Search.base.all
    proc {$ C} {Rainy C} end}}
```

```
{Browse {Search.base.all Snowy}}
```

# Exercises

76. Download SWI Prolog and Mozart 1.4.0 and install them in your laptop.
77. Execute the “`snowy(City)`” example. In Prolog, use “tracing” to follow backtracking step by step.
78. 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/Oz program for family relationships.