Logic Programming

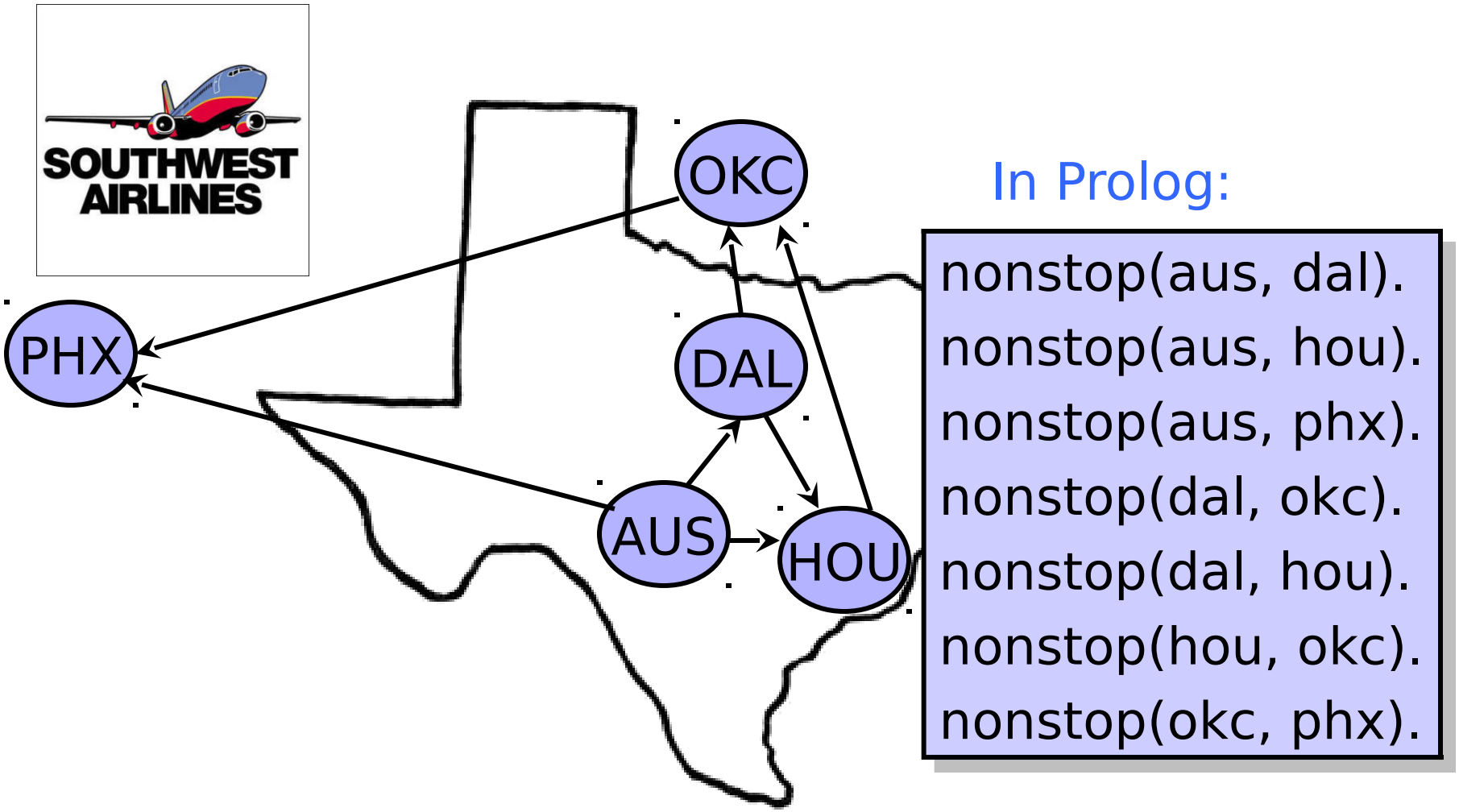
Syllabus

Lecture Series (hours)	Topics
1-4	Introduction and Motivation, Paradigms
5-10	Syntax and Semantics, BNF, Compilation
11-18	Data Types, Constructs, Functions, Activation Records, Names and Bindings
19-28	Concurrency, Lambda Calculus, Functional PLs, Logical PLs, Event driven programming
29-36	Virtual Machines, Managed Languages, JIT, Case study

Prolog

- Short for Programmation en logique
- Basic idea: the program declares the goals of the computation, not the method for achieving them
- Applications in AI, databases, even systems
  - Originally developed for natural language processing
  - Automated reasoning, theorem proving
  - Database searching, as in SQL
  - Expert systems
  - Recent work at Berkeley on declarative programming (describing logic but not the flow)

Example: Logical Database



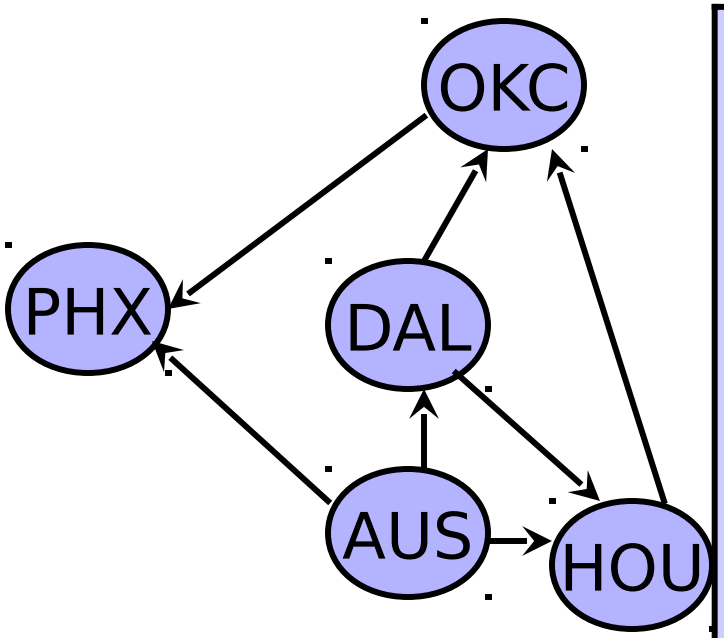
In Prolog:

```
nonstop(aus, dal).
nonstop(aus, hou).
nonstop(aus, phx).
nonstop(dal, okc).
nonstop(dal, hou).
nonstop(hou, okc).
nonstop(okc, phx).
```

Logical Database Queries

- Where can we fly from Austin?
- SQL
  - SELECT dest FROM nonstop WHERE source=“aus”;
- Prolog
  - ?- nonstop(aus, X).
  - More powerful than SQL because can use recursion

Flight Planning Example



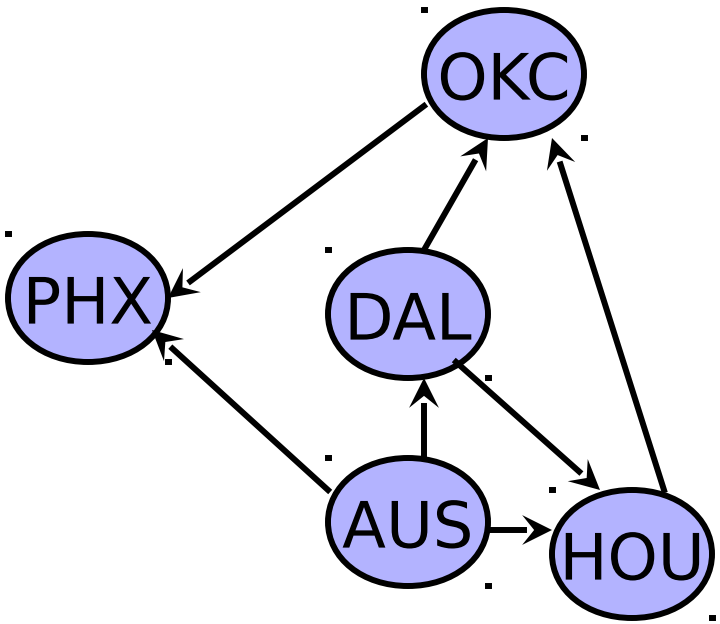

```
nonstop(aus, dal).
nonstop(aus, hou).
nonstop(aus, phx).
nonstop(dal, okc).
nonstop(dal, hou).
nonstop(hou, okc).
nonstop(okc, phx).
```

Each line is called a clause and represents a known fact

A fact is true if and only if we can prove it true using some clause

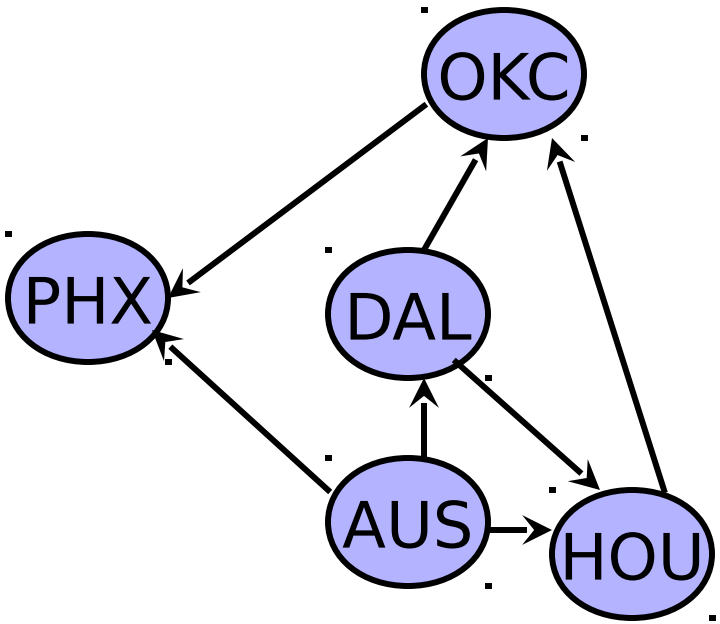
Relation: nonstop(X, Y) – there is a flight from X to Y

Queries in Prolog




```
?-nonstop(aus, dal).
Yes
?-nonstop(dal, okc).
Yes
?-nonstop(aus, okc).
No
?-
```

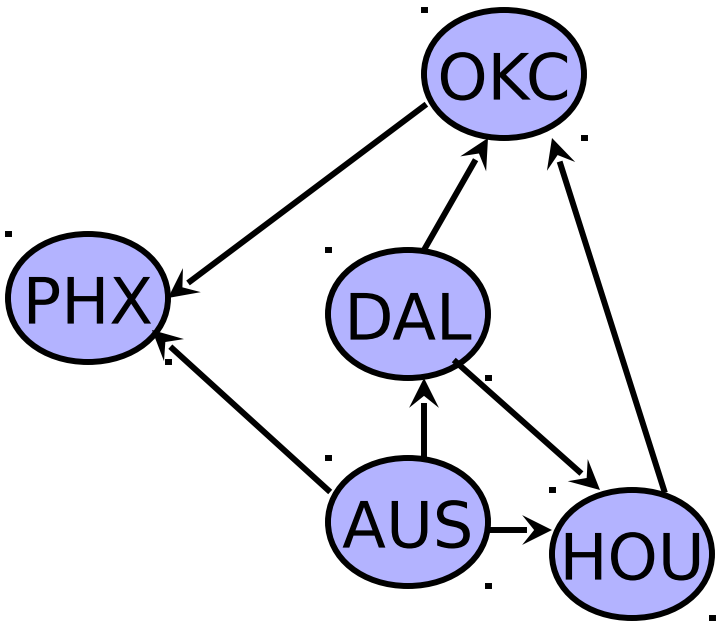
Logical Variables in Prolog



Is there an X such that nonstop(okc, X) holds?

```
?- nonstop(okc, X).
X=phx ;
No
?- nonstop(Y, dal).
Y=aus ;
No
?-
```

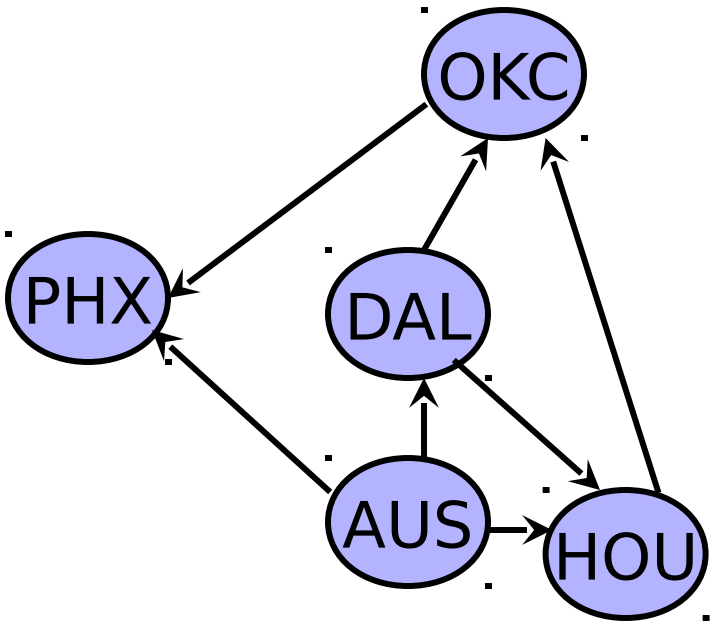
Non-Determinism




```
?- nonstop(dal, X).
X=hou ;
X=okc ;
No
?- nonstop(phx, X).
No
?-
```

Predicates may return multiple answers or no answers

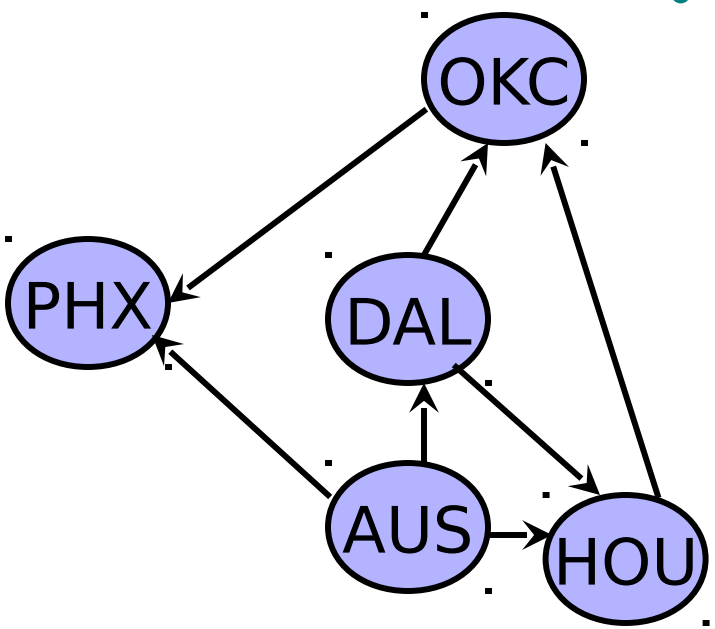
Logical Conjunction




```
?- nonstop(aus, X), nonstop(X, okc).
X=dal ;
X=hou ;
No
?-
```

Combine multiple conditions into one query

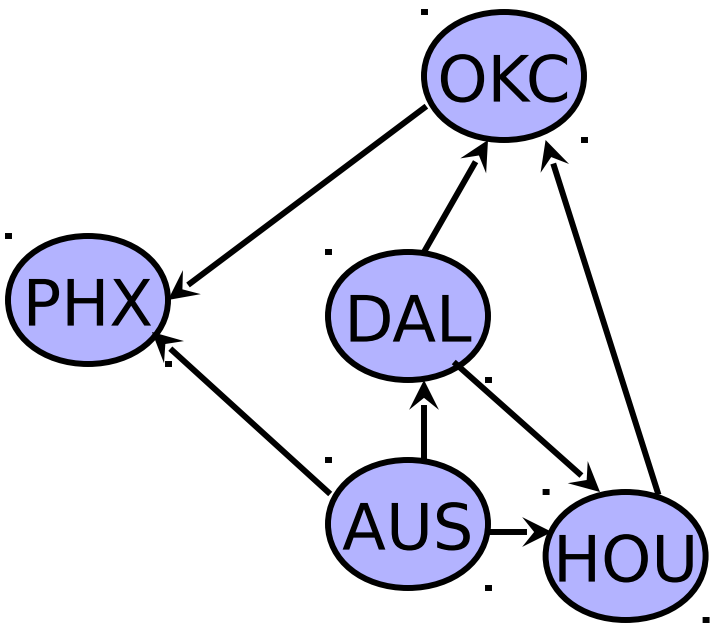
Derived Predicates



- Define new predicates using rules
- conclusion :- premises.
  - conclusion is true if premises are true

```
flyvia(From, To, Via) :-
        nonstop(From, Via),
        nonstop(Via, To).
?- flyvia(aus, okc, Via).
Via=dal ;
Via=hou ;
No
?-
```

Recursion



- Predicates can be defined recursively

```
reach(X, X).
reach(X,Z) :-
        nonstop(X, Y), reach(Y, Z).
```

```
?-reach(X, phx).
X=aus ;
X=dal ;
...
?-
```

Prolog Program Elements

- Prolog programs are made from terms
  - Variables, constants, structures
- Variables begin with a capital letter
  - Bob
- Constants are either integers, or atoms
  - 24, zebra, ‘Bob’, ‘.’
- Structures are predicates with arguments
  - n(zebra), speaks(Y, English)

Horn Clauses

- A Horn clause has a head h, which is a predicate, and a body, which is a list of predicates p1, p2, ..., pn
  - It is written as h ← p1, p2, ..., pn
  - This means, “h is true if p1, p2, ..., and pn are simultaneously true”
- Example
  - snowing(C) ← precipitation(C), freezing(C)
  - This says, “it is snowing in city C if there is precipitation in city C and it is freezing in city C”

Facts, Rules, and Programs

- A Prolog fact is a Horn clause without a right-hand side
  - Term.
    - The terminating period is mandatory
- A Prolog rule is a Horn clause with a right-hand side (:- represents ←)
  - term :- term1, term2, ... termn.
  - LHS is called the head of the rule
- Prolog program = a collection of facts and rules

Horn Clauses and Predicates

- Any Horn clause $h \leftarrow p1, p2, \ldots, pn$ can be written as a predicate $p1 \wedge p2 \wedge \ldots \wedge pn \supset h$, or, equivalently, $\neg(p1 \wedge p2 \wedge \ldots \wedge pn) \vee h$
- Not every predicate can be written as a Horn clause
  - Example: $literate(x) \supset reads(x) \vee writes(x)$

Answering Prolog Queries

- Computation in Prolog (answering a query) is essentially searching for a logical proof
- Goal-directed, backtracking, depth-first search
  - Resolution strategy:

    if h is the head of a Horn clause

    h ← terms

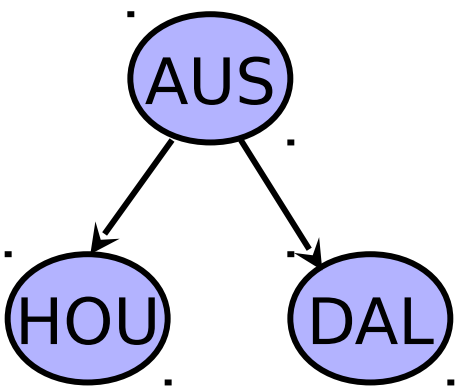
    and it matches one of the terms of another Horn clause

    t ← t1, h, t2

    then that term can be replaced by h's terms to form

    t ← t1, terms, t2
  - What about variables in terms?

Flight Planning Example



?- n(aus, hou).

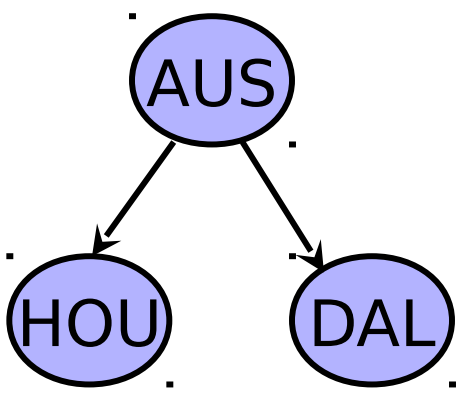
?- n(aus, dal).

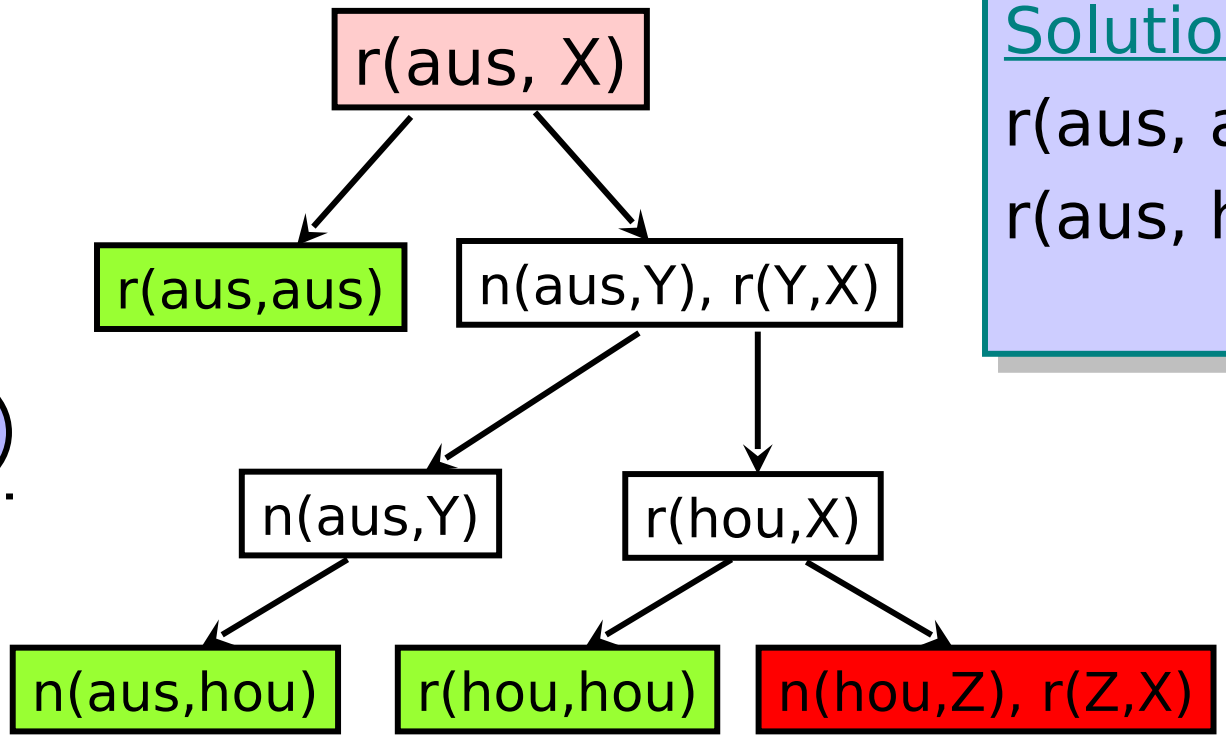
?- r(X, X).

?- r(X, Z) :- n(X, Y), r(Y, Z).

?- r(aus, X)

Flight Planning: Proof Search





Solution

r(aus, aus)

r(aus, hou)

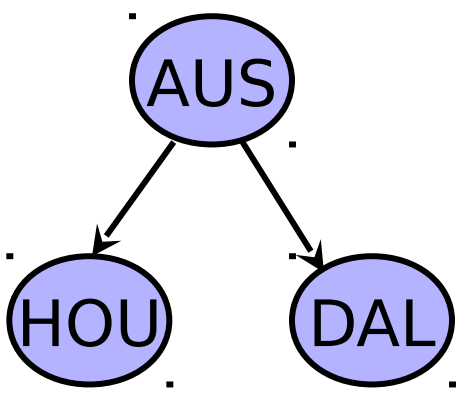
Rule 1:

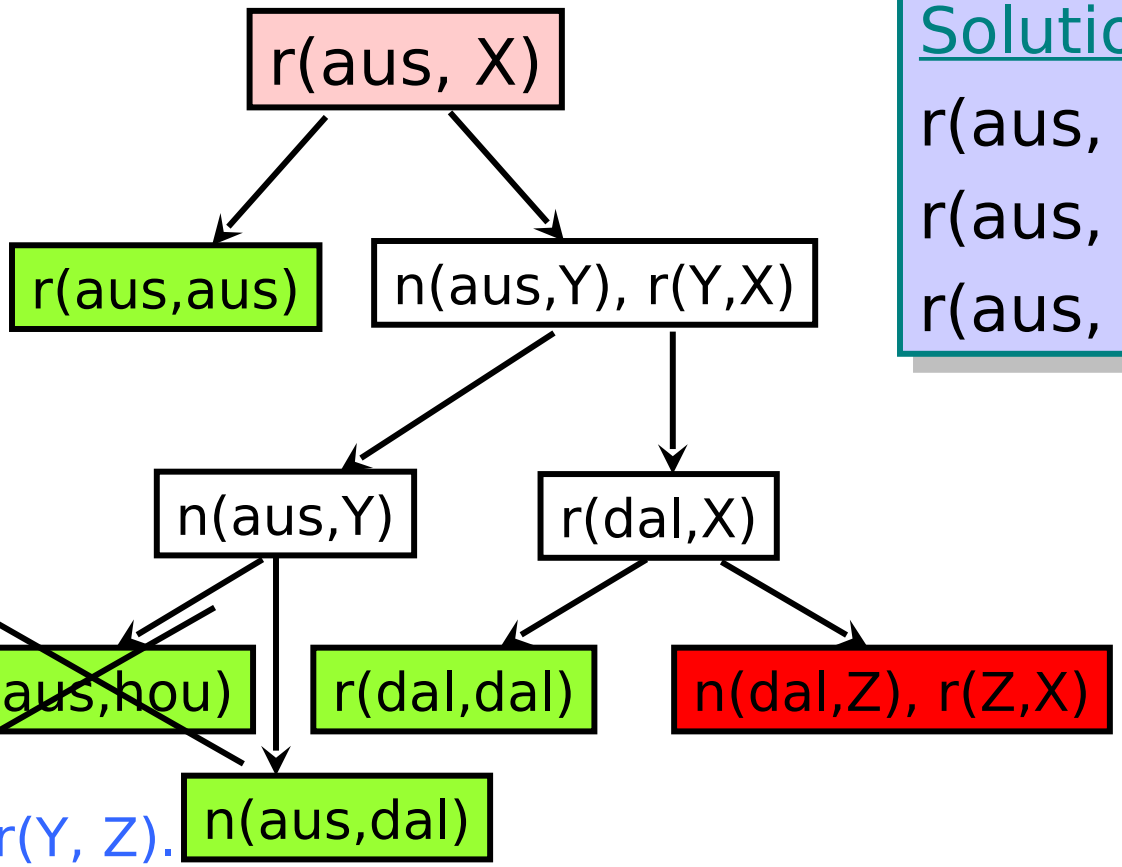
⇨ r(X, X).

Rule 2:

⇨ r(X, Z) :- n(X, Y), r(Y, Z).

Flight Planning: Backtracking





Solution

r(aus, aus)

r(aus, hou)

r(aus, dal)

Rule 1:

⇨ r(X, X).

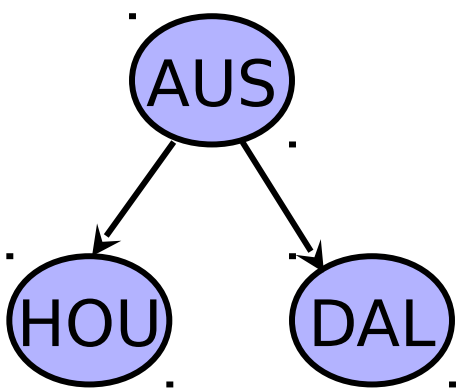
Rule 2:

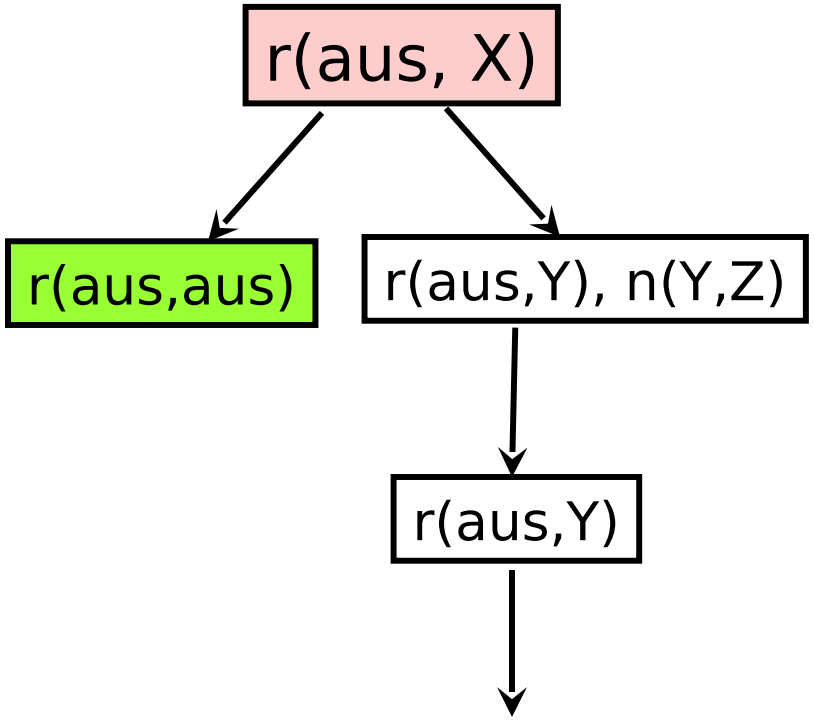
⇨ r(X, Z) :- n(X, Y), r(Y, Z).

Soundness and Completeness

- Soundness
  - If we can prove something, then it is logically true
- Completeness
  - We can prove everything that is logically true
- Prolog search procedure is sound, but incomplete (can go into infinite loops)

Flight Planning: Small Change





Solution

r(aus, aus)

Rule 1:

r(X, X).

Rule 2:

r(X, Z) :- r(X, Y), n(Y, Z).

instead of n(X,Y), r(Y,Z)

Infinite loop!

Syllabus

Lecture Series (hours)	Topics
1-4	Introduction and Motivation, Paradigms
5-10	Syntax and Semantics, BNF, Compilation
11-18	Data Types, Constructs, Functions, Activation Records, Names and Bindings
19-28	Concurrency, Lambda Calculus, Functional PLs, Logical PLs, Event driven programming
29-36	Virtual Machines, Managed Languages, JIT, Case study

What's a Scripting Language?

- Language used to write programs that compute inputs to another language processor
  - One language embedded in another
    - Embedded JavaScript computes HTML input to the browser
    - Shell scripts compute commands executed by the shell
- Common characteristics of scripting languages
  - String processing – since commands often strings
  - Simple program structure, define things "on the fly"
  - Flexibility preferred over efficiency, safety

Why JavaScript?

- "Active" web pages
- Web 2.0
  - AJAX, huge number of Web-based applications
- Some interesting and unusual features
  - First-class functions - interesting
  - Objects without classes - slightly unusual
  - Powerful modification capabilities - very unusual
    - Add new method to object, redefine prototype, …
- Many security and correctness issues

Common Uses of JavaScript

- Form validation
- Page embellishments and special effects
- Navigation systems
- Basic math calculations
- Dynamic content manipulation
- DOM – HTML elements
- CSS – Rules to tell browser how to display DOM
- JavaScript – Prog. Language, which manipulates DOM, CSS and does many dynamic things

Example 1: Add Two Numbers

```
<html>
    ...
 <p>  ... </p>
<script>
   var num1, num2, sum
   num1 = prompt("Enter first number")
   num2 = prompt("Enter second number")
   sum = parseInt(num1) + parseInt(num2)
   alert("Sum = " + sum)
</script>
   ...
</html>
```

Example 2: Browser Events

Mouse event causes page-defined function to be called

```
<script type="text/JavaScript">
    function whichButton(event) {
     if (event.button==1) {
            alert("You clicked the left mouse button!") }
     else {
            alert("You clicked the right mouse button!")
     }}
</script>
...
<body onmousedown="whichButton(event)">
...
</body>
```

Other events: onLoad, onMouseMove, onKeyPress, onUnLoad

Example 3: Page Manipulation

- Some possibilities
  - createElement(elementName)
  - createTextNode(text)
  - appendChild(newChild)
  - removeChild(node)
- Example: add a new list item

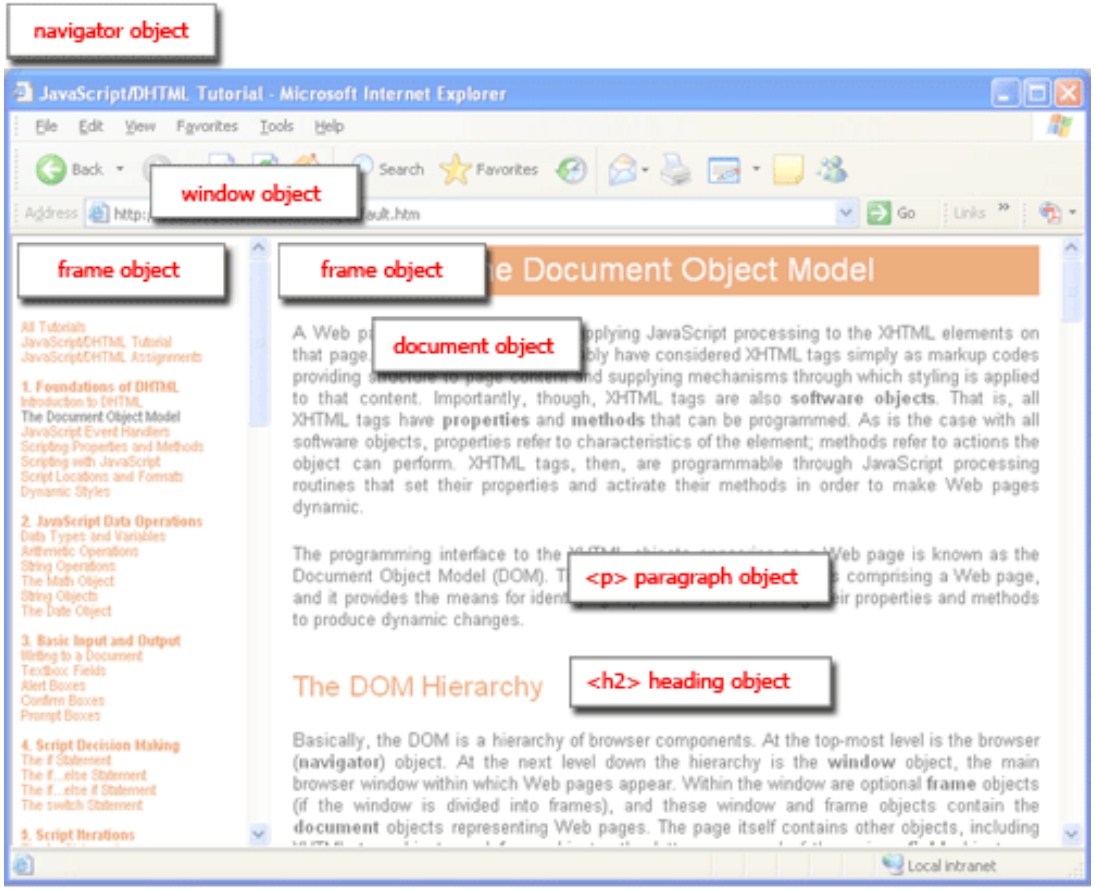
```
var list = document.getElementById('t1')
var newitem = document.createElement('li')
var newtext = document.createTextNode(text)
list.appendChild(newitem)
newitem.appendChild(newtext)
```

This uses the browser Document Object Model (DOM). We will focus on JavaScript as a language, not its use in the

Document Object Model (DOM)

- HTML page is structured data
- DOM provides representation of this hierarchy
- Examples
  - Properties: document.alinkColor, document.URL, document.forms[ ], document.links[ ], document.anchors[ ], …
  - Methods: document.write(document.referrer)
    - These change the content of the page!
- Also Browser Object Model (BOM)
  - Window, Document, Frames[], History, Location, Navigator (type and version of browser)

Browser and Document Structure



W3C standard differs from models supported in existing browsers

Reading Properties with JavaScript

Sample script

```
1. document.getElementById('t1').nodeName
2. document.getElementById('t1').nodeValue
3. document.getElementById('t1').firstChild.nodeName
4. document.getElementById('t1').firstChild.firstChild.nodeName
5. document.getElementById('t1').firstChild.firstChild.nodeValue
```

Sample HTML

```
<ul id="t1">
<li> Item 1 </li>
</ul>
```

- Example 1 returns "ul"
- Example 2 returns "null"
- Example 3 returns "li"
- Example 4 returns "text"
  - A text node below the "li" which holds the actual text data as its value
- Example 5 returns " Item 1 "

Language Basics

- JavaScript is case sensitive
  - onClick, ONCLICK, … are HTML, thus not case-sensitive
- Statements terminated by returns or semi-colons
  - x = x+1; same as x = x+1
- “Blocks” of statements enclosed in { …}
- Variables
  - Define using the var statement
  - Define implicitly by its first use, which must be an assignment
    - Implicit defn has global scope, even if occurs in nested scope!

JavaScript Blocks

- Use { } for grouping; not a separate scope

```
js> var x=3;
js> x
3
js> {var x=4; x}
4
js> x
4
```

- Not blocks in the sense of other languages

JavaScript Primitive Datatypes

- Boolean: true and false
- Number: 64-bit floating point
  - Similar to Java double and Double
  - No integer type
  - Special values NaN (not a number) and Infinity
- String: sequence of zero or more Unicode chars
  - No separate character type (just strings of length 1)
  - Literal strings using ' or " characters (must match)
- Special objects: null and undefined

Objects

- An object is a collection of named properties
- Think of it as an associative array or hash table
  - Set of name:value pairs
    - objBob = {name: “Bob", grade: 'A', level: 3};
  - Play a role similar to lists in Lisp / Scheme
- New members can be added at any time
    - objBob.fullname = 'Robert';
- Can have methods
- Can refer to this

Functions

- Functions are objects with method called “( )”
  - A property of an object may be a function (=method)
    - function max(x,y) { if (x>y) return x; else return y;};
    - max.description = “return the maximum of two arguments”;
  - Local declarations may appear in function body
- Call can supply any number of arguments
  - functionname.length : # of arguments in definition
  - functionname.arguments.length : # arguments in call
  - Basic types are passed by value, objects by reference
- “Anonymous” functions
  - (function (x,y) {return x+y}) (2,3);

Examples of Functions

- Curried functions
  - function CurriedAdd(x) { return function(y){ return x+y} };
  - g = CurriedAdd(2);
  - g(3)
- Variable number of arguments
  - function sumAll() {

```
var total=0;
for (var i=0; i< sumAll.arguments.length; i++)
    total+=sumAll.arguments[i];
return(total); }
```

  - sumAll(3,5,3,5,3,2,6)

Anonymous Functions

- Anonymous functions very useful for callbacks
  - setTimeout(function() { alert("done"); }, 10000)
  - Evaluation of alert("done") delayed until function call
- Simulate blocks by function definition and call
  - var u = { a:1, b:2 }
  - var v = { a:3, b:4 }
  - (function (x,y) {

        var tempA = x.a; var tempB =x.b; // local variables

        x.a=y.a; x.b=y.b;

        y.a=tempA; y.b=tempB

    }) (u,v)   // Works because objs are passed by ref

Basic Object Features

- Use a function to construct an object
  - function car(make, model, year) {
    this.make = make;
    this.model = model;
    this.year = year; }
- Objects have prototypes, can be changed
  - var c = new car(“Ford”,”Taurus”,1988);
  - car.prototype.print = function () {
    return this.year + “ “ + this.make + “ “ + this.model;}
  - c.print();

JavaScript in Web Pages

- Embedded in HTML page as <script> element
  - JavaScript written directly inside <script> element
    - <script> alert("Hello World!") </script>
  - Linked file as src attribute of the <script> element

    <script type="text/JavaScript" src="functions.js"></script>
- Event handler attribute

  <a href="http://www.yahoo.com "
  onmouseover="alert('hi');">
- Pseudo-URL referenced by a link

  Click me

We are looking at JavaScript as a language; ignore BOM, DOM, AJAX

Garbage Collection

- Automatic reclamation of unused memory
- Navigator 2: per-page memory management
  - Reclaim memory when browser changes page
- Navigator 3: reference counting
  - Each memory region has associated count
  - Count modified when pointers are changed
  - Reclaim memory when count reaches zero
- Navigator 4: mark-and-sweep, or equivalent
  - Garbage collector marks reachable memory
  - Sweep and reclaim unreachable memory

Syllabus

Lecture Series (hours)	Topics
1-4	Introduction and Motivation, Paradigms
5-10	Syntax and Semantics, BNF, Compilation
11-18	Data Types, Constructs, Functions, Activation Records, Names and Bindings
19-28	Concurrency, Lambda Calculus, Functional PLs, Logical PLs, Event driven programming
29-36	Virtual Machines, Managed Languages, JIT, Case study