Introductions

- Ruben Safir MS Computer Sciences Candidate
- Artifical Intelligence
- Long Island University Brooklyn
- Fall 2015
- Professor Ping-Tsai Chung
- Expert Systems with Clips and Free Software
- Catching Fish with a the Rod and Reel in NYC
- Source Documentation http://www.nylxs.com/docs/grad_school/artificial_intelligence/fishing_project/

Problem Description

- Daily trips are made out of Sheepshead Bay Brooklyn, City Island, and Marinas around NYC to take passengers for recreational fishing trips on the local waters.
- City, State and Federal Laws regulate which fish can be targeted, and brought home with limitations on size, numbers, and time of season when one can fish.
- Jurisdictional issues make it necessary to determine when fishing in Federal or State regulated waters is allowed.
- In addition to legal regulations, fish enter our waters on seasonal runs. Some fish come when it is cold, others prefer warm water. Fish can be found in different waters at different times.
- Some fish are more popular than others, either for the taste or the fish, or the fun in catching. Some seasons generate excitement for fisherman in local waters, especially tautog and striped bass.
- Captains of fishing boats acquire and share data and hot tips, and otherwise keep certain secret spots to themselves, something that is increasingly difficult in the modern surveillance state.
- In order to remain profitable, Fishing Boat Captains have to juggle a sizable amount of expert information to remain in business, and then boost sales with well placed reports in the fishing press and among local tackle shops and places fishermen gather.

When it comes to fishing EVERYONE wants the inside scoop, making an expert system, perhaps very useful.

Tataug or Blackfish are the tastiest fish in the local waters and tricky to catch

Blue Fish World Migrants - around all year - good fighters, not that tasty

Cod Fish - Staple of Generations of East Coast Households. Numbers greatly diminished and now regulated and even closed in New England

Black Sea Bass: Perhaps the best eating fish in local waters. Very abundant in NY due to conservation efforts, but under close watch by the federal government, especially in southern states. I've seen them sold for $22 a pound in the Union Square Green Market.

Fluke - Very popular, and live close to shore making them inexpensive to fish for when in season.

Striped bass are the queens of the NY Fishing scene. They fill a niche similar to Salmon in other places, traveling up and down the Hudson River, and Chesapeake Bay is large runs. Not legal to keep in Federal Waters. They are closely watched and currently have a limit of one fish a trip, over 28 inches long. This was the fall run a few seasons ago. When news gets out that Stripers are on the run, boats are filled and everyone is happy for a few days.

A prized catch of Striped Bass that fed the family for a few days.

Scrup - un fish to catch and the backbone of the NY recreational fishery

When scup are around, they are fun to catch and fill the buckets

Along with these, I don't cover popular fish that are rarer in our harbor, but still popular to include Winter Flounder, Lemon Sole, Herring, Mackerel, Weakfish, and a variety of Tuna which show up from time to time.

Forward Chaining Methodology and Graph

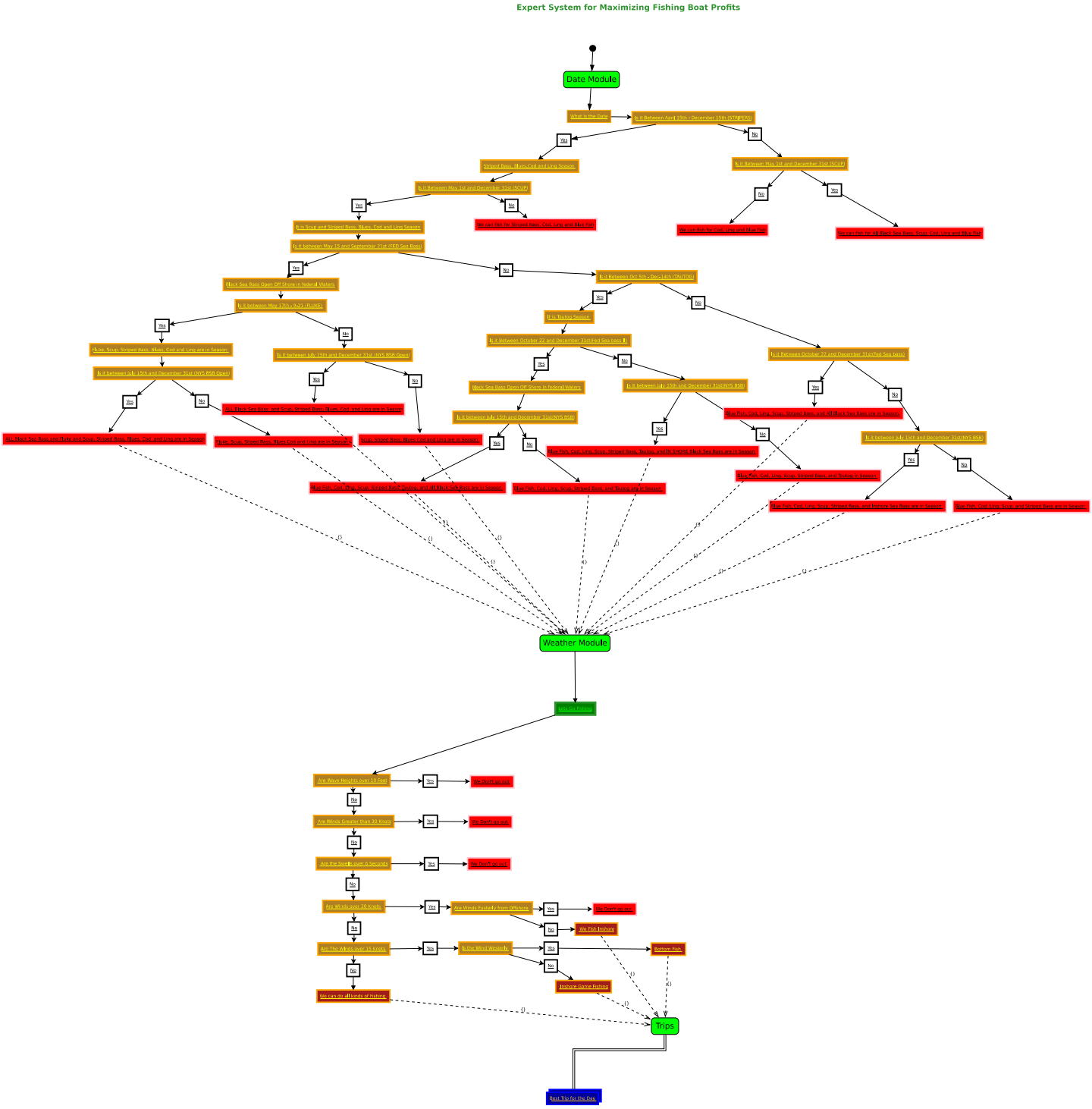
- Since we know the facts which affect our decisions with a fair bit of precision, we would seem to have an good condition for forward chaining of our facts to draw conclusions.
- Fishing Season are set out by law and can be exactly defined. However, overlapping calenders for different species can make a decision tree difficult to create. A Schedule looks like this:

Marine Recreational Fishing Limits

Species	Size Limits (Total Length in Inches)*	Possession Limits (Number of Fish)	Open Seasons
Summer flounder (fluke)^	18	5	May 17 - Sept 21
Winter flounder	12	2	April 1 - May 30
Tautog (blackfish)	16	4	Oct 5 - Dec 14
Bluefish (including "snappers")	No minimum size for first 10 fish 12 TL for the next 5	15 No more than 10 of which shall be less than 12" TL	All year
Black Sea Bass	14	8 10	July 15 - Oct 31 Nov 1 - Dec 31
Scup (Porgy)	10	30	May 1 - Dec 31
Scup (Porgy) *Anglers aboard licensed party/charter boats*	10	30 45	May 1 - Dec 31 Sept 1 - Oct 31
Yellowtail flounder	13	No limit	All year

Reducing this to a binary tree is very difficult. Resulting leafs for any date allows for the fishing of multiple species of fish. And that is before weather conditions are considered. As a result, we need to use three modules of expert system rules to accomplish this.

This is the resulting decision tree constructed from the Free Software program DIA which is available at http://dia-installer.de/index.html.en and licensed under the GPL2 software license. Downloads in anonymity is allowed and encouraged for as long as the Government or Watson allows. See the decision tree diagram on the full next page.


Expert System for Maximizing Fishing Boat Profits
Date Module
Weather Module
Trips
Boat Trip for the Day

December 2015 | Artificial Intelligence: Expert Systems with CLIPS and Free Software | LIU/Brooklyn Artificial Intelligence

Fishing Season Schedule strategy

- In evaluating the schedules for fishing seasons it becomes apparent very quickly how unwieldy this data tree and the resulting code base will become, even with just a moderate number of species, and only two jurisdictions. If I add neighboring states, and the mostly ignored reciprocity laws (what happens when a New York boat fishes in New Jersey waters) it can take a team of very smart people to create the correct decision tree. Fortunately, with the use of the inference engine in CLIPS, we can do things in a slightly different way.
- There are two ways to approach this decision tree. First and most obvious is to handle it like a series of nodes, each one triggering the next according to response. CLIPS is an inference engine and such a solution is suggested by Giarratano & Riley in there text on CLIPS, *"Expert Systems, Principles and Programming"*. In this case, rules are created where the user is prompted to give an answer questions which cause the CLIPS agenda to start the next appropriate rule until a leaf is reaches with an answer. A quick mock up of this kind of code is as follows:

```
(deftemplate node
        (slot name )
        (slot type )
        (slot question )
        (slot yes-node)
        (slot no-node )
        (slot answer )
)
```

This is similar to what we would see as a structure in procedural code.

```
(deffacts tree
        (node (name root)(type decision)
                (question "Is the animal warm blooded?"  )
                (yes-node node1) (no-node node2))
        (node (name node1)(type decision)
                (question "Does it purr?")
                (yes-node node3) (no-node node4))
        (node (name node2) (type answer) (answer snake))
        (node (name node3) (type answer) (answer cat))
        (node (name node4) (type answer) (answer dog))
        (current-node root)
)
```

These create nodes on reset that either push you down the tree (decisions) or are leafs (answer). Now we just need some rules to stitch it together.

```
(defrule ask-decision-node-question
        ?node <- (current-node ?name)
        (node (name ?name)
                        (type decision)
                        (question ?question))
                        (not (answer ?))
=>
        (printout t ?question "(yes or no) ")
        (assert (answer (read)))
)
;*******
;******
;*****
(defrule bad-answer
        ?answer <- (answer ~yes&~no)
=>
        (retract ?answer)
)
```

```
(defrule proceed-to-yes-branch
        ?node <- (current-node ?name)
        (node (name ?name)
                        (type decision)
                        (yes-node ?yes-branch))
   ?answer <- (answer yes)
  =>
        (retract ?node ?answer)
        (assert (current-node ?yes-branch))
)
;*******
;******
;*****

(defrule proceed-to-no-branch
        ?node <- (current-node ?name)
        (node (name ?name)
                        (type decision)
                        (no-node ?yes-branch))
   ?answer <- (ansert no)
  =>
        (retract ?node ?answer)
        (assert (current-node ?no-branch))
)
```

One set of rules for yes decisions, another for no decisions and a third bad answers. The last rule is for the leaf

```
(defrule correct
        ?node <- (current-node ?name)
        (node (name ?name) (tyoe answer)(answer ?value))
        (not (answer ?))
                                        =>
        (prinout t "I guess it is a " ?value crlf)
)
```

At this point I note three things here:

1. This code is an adaptation of the code example in the Giarratano & Riley text noted above.
2. I have not tested this code and leave that to you.
3. The complete code in the text is far boarder than this and worth looking into and testing in its own right. I encourage you to get a copy of the text.

- As we examine this, we can see that this will allow us to create our tree exactly like our diagram, but in this case, it is a waste of a good inference engine. It assumes a single species result. We have results of several species at once and then each of those branches need to be tested for all possible weather conditions, which is a lot of repeating code, and code that repeats itself is a clear indication that the approach to the problem is not correct.
- **So we have a better approach**

```
/* function to open any folds
containing a jumped-to line
before jumping to it */
function JumpToLine()
{
var lineNum;
lineNum =
window.location.hash;
lineNum = lineNum.substr(1);
/* strip off '#' */

if (lineNum.indexOf('L') ==
-1) {
lineNum = 'L'+lineNum;
}
lineElem =
document.getElementById(line
Num);
/* Always jump to new
location even if the line was
hidden inside a fold, or
* we corrected the raw
number to a line ID.
*/
if (lineElem) {
lineElem.scrollIntoView(true
);
}
return true;
}
if ('onhashchange' in window)
{
window.onhashchange =
JumpToLine;
}

-->December 2015
```

Systems with CLIPS and Free Software

Intelligence

Let Clips do the work!!

- If we structure our facts correctly we can let CLIP automatically generate the binary tree for us.

```
 2 (defmodule FISH (export deftemplate menu) )
 3 (deftemplate FISH::fish
 4    (slot species )
 5    (multislot habitat )
 6    (slot seasonal_availability ) ;date of fish runs
 7    (slot open_season ) ;the date legal open season begins
 8    (slot close_season ) ;the date the legal season closes
 9    (slot size_min ) ;minimum legal size allowed to be taken
10    (slot limit ) ;legal maximum allowed number of fish allowed
11    (slot fishing_type ) ;bottom or gamefishing
12    (slot fishing_location ) ;inshore or offshore
13    (multislot best_trip ) ; best to fish day or night
14    (slot desireability) ; how much people like to fish
15 )
```

- Presented here is the new backbone of out clips program. Instead of a new node for every combination of facts, we let the CLIPS inference engine do what was designed to do. We move the attributes of species of fish to the fish facts themselves, including the open and closed seasons. Now we can quickly scale up the program, and change facts rapidly as would be needed as seasons change with every new year, and can expand our program quickly.
- With CLIPS, each combination of facts makes a NEW fact. So we have 12 attributes, so if I have blue fish inshore and blue fish off shore, I can list both. If I have porgies with one set of rules in January and another in July, I can put both in my list of facts.
- In total I defined 28 different variations of fish species, which is far better than the possibly hundreds of combinations of nodes I would have if I didn't move the fish attributes to the fish facts.

```
60 (deffacts FISH::fishes
61    (fish (species tautog) (habitat bottom wrecks) (seasonal_availability winter_tautog )
62          (open_season 21022) (close_season 21231 ) (size_min 24 )
63          (limit 4 ) (fishing_type bottom ) (fishing_location offshore )
64          (best_trip day ) (desireability 5)
65    )
66    (fish (species tautog) (habitat bottom wrecks) (seasonal_availability winter_tautog )
67          (open_season 21022) (close_season 21231 ) (size_min 24 )
68          (limit 4 ) (fishing_type bottom ) (fishing_location offshore_federal_waters )
69          (best_trip day ) (desireability 5)
70    )
71    (fish (species tautog) (habitat bottom wrecks) (seasonal_availability winter_tautog )
72          (open_season 21022) (close_season 21231 ) (size_min 24 )
73          (limit 4 ) (fishing_type bottom ) (fishing_location inshore )
74          (best_trip day ) (desireability 2)
75    )
76    (fish (species striped_bass) (habitat inshore current rocks ) (seasonal_availability
spring_bass_run )
77          (open_season 20101 ) (close_season 21231 ) (size_min 31  )
78          (limit 1 ) (fishing_type game) (fishing_location inshore )
79          (best_trip day night ) (desireability 4)
80    )
81    (fish (species striped_bass) (habitat inshore current rocks ) (seasonal_availability
fall_bass_run )
82          (open_season 20101) (close_season 21231 ) (size_min 31 )
83          (limit 1 ) (fishing_type game ) (fishing_location inshore )
84          (best_trip day night ) (desireability 4)
85    )
86    (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
87          (open_season 20501 ) (close_season 20831 ) (size_min 10 )
88          (limit 30 ) (fishing_type bottom ) (fishing_location inshore )
89          (best_trip day ) (desireability 3 )
90    )
91    (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
92          (open_season 20501 ) (close_season 20831 ) (size_min 10 )
93          (limit 30 ) (fishing_type bottom ) (fishing_location offshore )
94          (best_trip day ) (desireability 4 )
95    )
96    (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
97          (open_season 20501 ) (close_season 20831 ) (size_min 10 )
98          (limit 30 ) (fishing_type bottom ) (fishing_location offshore_federal_waters )
99          (best_trip day ) (desireability 4 )
```

```
100     )
101     (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
102           (open_season 20901 ) (close_season 21031 ) (size_min 10 )
103           (limit 45 ) (fishing_type bottom ) (fishing_location offshore )
104           (best_trip day night) (desireability 4 )
105     )
106     (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
107           (open_season 20901 ) (close_season 21031 ) (size_min 10 )
108           (limit 45 ) (fishing_type bottom ) (fishing_location offshore_federal_waters )
109           (best_trip day night) (desireability 4 )
110     )
111     (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
112           (open_season 20901 ) (close_season 21031 ) (size_min 10 )
113           (limit 45 ) (fishing_type bottom ) (fishing_location inshore )
114           (best_trip day night) (desireability 3 )
115     )
116     (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
117           (open_season 21101 ) (close_season 21231 ) (size_min 10 )
118           (limit 30 ) (fishing_type bottom ) (fishing_location inshore )
119           (best_trip day ) (desireability 3 )
120     )
121     (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
122           (open_season 21101 ) (close_season 21231 ) (size_min 10 )
123           (limit 30 ) (fishing_type bottom ) (fishing_location offshore )
124           (best_trip day ) (desireability 4 )
125     )
126     (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
127           (open_season 21101 ) (close_season 21231 ) (size_min 10 )
128           (limit 30 ) (fishing_type bottom ) (fishing_location offshore_federal_waters )
129           (best_trip day ) (desireability 4 )
130     )
131     (fish (species fluke ) (habitat bottom ) (seasonal_availability warm_months)
132           (open_season 20517) (close_season 20921 ) (size_min 18 )
133           (limit 5 ) (fishing_type bottom ) (fishing_location inshore )
134           (best_trip day ) (desireability 3)
135     )
136     (fish (species bluefish) (habitat open_water current ) (seasonal_availability all_year)
137           (open_season 20101 ) (close_season 21231 ) (size_min 0 )
138           (limit 500 ) (fishing_type game ) (fishing_location inshore )
139           (best_trip day ) (desireability 2 )
140     )
141     (fish (species bluefish) (habitat open_water current ) (seasonal_availability all_year)
142           (open_season 20101 ) (close_season 21231 ) (size_min 12 )
143           (limit 15 ) (fishing_type game ) (fishing_location offshore )
144           (best_trip day ) (desireability 3 )
145     )
146     (fish (species bluefish) (habitat open_water current ) (seasonal_availability all_year)
147           (open_season 20101 ) (close_season 21231 ) (size_min 12 )
148           (limit 15 ) (fishing_type game ) (fishing_location offshore_federal_waters )
149           (best_trip day ) (desireability 3 )
150     )
151     (fish (species cod ) (habitat open_water) (seasonal_availability winter )
152           (open_season 20101 ) (close_season 21231 ) (size_min 22 )
153           (limit 10 ) (fishing_type bottom ) (fishing_location offshore )
154           (best_trip day ) (desireability 3)
155     )
156     (fish (species cod ) (habitat open_water) (seasonal_availability winter )
157           (open_season 20101 ) (close_season 21231 ) (size_min 22 )
158           (limit 10 ) (fishing_type bottom ) (fishing_location offshore_federal_waters )
159           (best_trip day ) (desireability 3)
160     )
161     (fish (species black_sea_bass ) (habitat bottom ) (seasonal_availability warm_months )
162           (open_season 20715) (close_season 21031 ) (size_min 14 )
163           (limit 8 ) (fishing_type bottom ) (fishing_location inshore )
164           (best_trip day ) (desireability 3 )
```

```
165    )
166    (fish (species black_sea_bass ) (habitat bottom ) (seasonal_availability warm_months )
167          (open_season 21101) (close_season 21231 ) (size_min 14 )
168          (limit 10 ) (fishing_type bottom ) (fishing_location inshore )
169          (best_trip day ) (desireability 3 )
170    )
171    (fish (species black_sea_bass ) (habitat bottom ) (seasonal_availability warm_months )
172          (open_season 20715) (close_season 21031 ) (size_min 14 )
173          (limit 8 ) (fishing_type bottom ) (fishing_location offshore )
174          (best_trip day ) (desireability 3 )
175    )
176    (fish (species black_sea_bass ) (habitat bottom ) (seasonal_availability warm_months )
177          (open_season 21101) (close_season 21231 ) (size_min 14 )
178          (limit 10 ) (fishing_type bottom ) (fishing_location offshore )
179          (best_trip day ) (desireability 3 )
180    )
181    (fish (species black_sea_bass_federal ) (habitat bottom ) (seasonal_availability
warm_months )
182          (open_season 20515) (close_season 20921 ) (size_min 14 )
183          (limit 10 ) (fishing_type bottom ) (fishing_location offshore_federal_waters )
184          (best_trip day ) (desireability 3 )
185    )
186    (fish (species black_sea_bass_federal ) (habitat bottom ) (seasonal_availability
warm_months )
187          (open_season 21022) (close_season 21231 ) (size_min 14 )
188          (limit 10 ) (fishing_type bottom ) (fishing_location offshore_federal_waters )
189          (best_trip day ) (desireability 3 )
190    )
191    (fish (species ling ) (habitat mud bottom ) (seasonal_availability winter )
192          (open_season 20101) (close_season 21231 ) (size_min  0)
193          (limit 500 ) (fishing_type bottom ) (fishing_location offshore )
194          (best_trip day ) (desireability 2)
195    )
196    (fish (species ling ) (habitat mud bottom ) (seasonal_availability winter )
197          (open_season 20101) (close_season 21231 ) (size_min  0)
198          (limit 500 ) (fishing_type bottom ) (fishing_location offshore_federal_waters )
199          (best_trip day ) (desireability 2)
200    )
201 )
```

- Now all I need to do it prompt the user for a date, and I can let clips pull out all the fish I can fish for on that date, by comparing the opening an close dates for the season. The only problem is that CLIPS has no date data type. So that leaved me to make a function that would test dates and return true. This was done as follows:

```
 29 (deftemplate FISH::season
 30    (slot description ) (slot start_date) (slot end_date)
 31 )
 32 (deftemplate FISH::date
 33     (slot categ (type SYMBOL) (allowed-symbols today start_date end_date) )
 34     (slot day (type INTEGER) (range 1 31))
 35     (slot month (type INTEGER) (range 1 12))
 36     (slot dateint (type INTEGER)(range 20101 21231 )) ; (dateint  - 20000 ) is the date
 37 )
 38
 39 (deffunction FISH::openseason (?start ?end ?today )
 40    (if (> ?start ?end )
 41     then
 42      (or
 43          (and (>= ?today ?start ) (<= ?today  21231))
 44          (and (>= ?today 20101 ) (<= ?today ?end))
 45       )
 46      else (and (>= ?today ?start) (<= ?today ?end))
 47    )
 48 )
```

```
 7 (defrule FISH::getdate
 8    =>
 9  ( printout t crlf crlf crlf crlf crlf crlf crlf crlf crlf crlf )
10  ( printout t crlf crlf crlf crlf crlf crlf crlf crlf crlf crlf )
11  ( printout t crlf crlf crlf crlf crlf crlf crlf crlf crlf crlf )
12  ( printout t crlf crlf crlf crlf crlf crlf crlf crlf crlf crlf )
13  ( printout t crlf crlf crlf crlf crlf crlf crlf crlf crlf crlf )
14  ( printout t crlf "Enter the Date" crlf "What is the Month (as an integer) ==>")
15  (bind ?month (read) )
16  (printout t crlf "What is the Day (as an integer)==> ")
17  (bind ?day (read) )
18  (bind ?day2 (+ ?day  10000))
19  (bind ?month2 (+ (* ?month  100) 10000))
20  (assert (date (categ today) (month ?month) (day ?day) (dateint (+ ?month2 ?day2))))
21 )
```

- The rule getdate does the math needed to create an integer which can be compared in the function openseason. Poof and gone is a huge unwieldy decision tree, and instead CLIPS does the job for us.


```
/* function to open any folds containing a jumped-to line before jumping to it */
function JumpToLine()
{
var lineNum;
lineNum = window.location.hash;
lineNum = lineNum.substr(1); /* strip off '#' */

if (lineNum.indexOf('L') == -1) {
lineNum = 'L'+lineNum;
}
lineElem = document.getElementById(lineNum);
/* Always jump to new location even if the line was hidden inside a fold, or
 * we corrected the raw number to a line ID.
*/
if (lineElem) {
lineElem.scrollIntoView(true);
```

```
                }
            return true;
                }
if ('onhashchange' in window)
                {
    window.onhashchange =
          JumpToLine;
                }

      -->December 2015
```

.

Full Overview - how is the weather and can we make a few bucks.

- With a basic understanding, we can now look at the full code. There are a few things that the novice CLIPS programmer (such as myself) needs to understand.
    - At its heart and soul, CLIPS is a huge regular expression engine. In many regards, using it reminds me Unix power tools like GAWK and even Perl. The top of every rule, CLIPS defrules has a pattern that has to be matched according to a specific set of, not nearly obvious, rules.
    - On the other hand, CLIP leans heavily on the functional language paradigm. I think that the authors of CLIP view its shell like many LISP coders view EMACS, but EMACS is a better editor.
    - Getting started with CLIPS is easy. Learning it well, that can take a long time. It is a fickle programming language and having a good programming text editor is vital to spotting the inevitable lost parenthesis.
    - Good programming languages shape the way that you thing. CLIPS does this. After knocking your head around with it for a while, you begin to see its roots in the Artificial Intelligence area. Standard textbooks become easier to read and understand. And new possible solutions to problems arise as you develop a sense of what it does.
    - Every TIME you assert a fact into a scope, the entire scope is reevaluated in full. You can't say this enough.
    - Every TIME you assert a fact into a scope, the entire scope is reevaluated in full. You can't say this enough.
    - Every TIME you assert a fact into a scope, the entire scope is reevaluated in full. You can't say this enough.
    - Every TIME you assert a fact into a scope, the entire scope is reevaluated in full. You can't say this enough. *Learning CLIPS means learning how its engine works.*
    - CLIPS has the ability to create modules that allow for results to be passed from one scope to another. I leveraged this ability. Without that ability, I would have fallen back on procedural programming techniques that for the purposes of this project, was not my goal. We have four modules: FISH, WEATHER, TRIP, and REPORT
- Source Code Links:

  fish2.clp : The main program that loads everything else and runs. You can run it in GNU/Linux with the command `clips -f2 ./fish2.dat`

```
ruben@localhost:~/bin/clips
*********Best Trip**********
Trip: offshore_federal_waters   Method: bottom  Species: tautog scup cod black_sea_bass_federal ling
*********Best Trip**********


---------------------------------------------------------------------------
Trip: offshore_federal_waters   Method: bottom  Species: tautog scup cod black_sea_bass_federal ling
Profit Points:   19

Trip: offshore_federal_waters   Method: game    Species: bluefish
Profit Points:    3

Trip: offshore                  Method: bottom  Species: tautog scup cod black_sea_bass ling
Profit Points:   19

Trip: offshore                  Method: game    Species: bluefish
Profit Points:    3

Trip: inshore                   Method: bottom  Species: tautog scup black_sea_bass
Profit Points:   10

Trip: inshore                   Method: game    Species: striped_bass striped_bass bluefish
Profit Points:   12

Trip: offshore                  Method: bottom  Species: tautog scup cod black_sea_bass ling
*********Best Trip**********


---------------------------------------------------------------------------
MAX Points is ==>19
*********Best Trip**********
         CLIPS (Quicksilver Beta 5/31/08)
CLIPS>
```

- The fishing program uses a scoring system to evaluate which would be the best trip to take on a given day under a give set of weather conditions. Each species has a point assigned from 1-5 of desirability. If a trip can target more than one species, good! Then the scores for each species is added up. The best trip is notated and all alternatives are also listed.

- The program is built with expansion in mind, so those extra facts you see are not useless. They just represent features not implemented.
- Bonus seasons are implemented. So striped bass get a bump during the likely times of the fall and spring runs. the code works, but I would like to make it work a little better.
- Areas of expansion would include a module for bonus points for hot news. We should have bonus points for point good reports on the internet which will drive sales. One might add daily results and crew reports to make bonus points for different captains. Not all captains are the same. GIS and mapping information, which is currently built into postgres, is also possible with some work. We can also include a section for bait, what are the fish biting today? Weather can be expaned. Porgies don't bite when it rains more than 4 inches. Tautog don't bite until water temperature drops below 56 F. So this topic can include a lot more work.

```
  <!--

/* function to open any folds containing a jumped-to line before jumping to it */
function JumpToLine()
{
  var lineNum;
  lineNum = window.location.hash;
  lineNum = lineNum.substr(1); /* strip off '#' */

  if (lineNum.indexOf('L') == -1) {
    lineNum = 'L'+lineNum;
  }
  lineElem = document.getElementById(lineNum);
  /* Always jump to new location even if the line was hidden inside a fold, or
   * we corrected the raw number to a line ID.
   */
  if (lineElem) {
    lineElem.scrollIntoView(true);
  }
  return true;
}
if ('onhashchange' in window) {
  window.onhashchange = JumpToLine;
}

-->  1 (load fish.dat)
    2 (load fish2.dat)
    3 (load fish3.dat)
    4 (load fish4.dat)
    5 (reset)
    6 (focus FISH)
    7 (defrule FISH::getdate
    8    =>
    9  ( printout t crlf crlf crlf crlf crlf crlf crlf crlf crlf crlf )
   10  ( printout t crlf crlf crlf crlf crlf crlf crlf crlf crlf crlf )
   11  ( printout t crlf crlf crlf crlf crlf crlf crlf crlf crlf crlf )
   12  ( printout t crlf crlf crlf crlf crlf crlf crlf crlf crlf crlf )
   13  ( printout t crlf crlf crlf crlf crlf crlf crlf crlf crlf crlf )
   14  ( printout t crlf "Enter the Date" crlf "What is the Month (as an integer)
==>")
   15  (bind ?month (read) )
   16  (printout t crlf "What is the Day (as an integer)==> ")
   17  (bind ?day (read) )
   18  (bind ?day2 (+ ?day  10000))
   19  (bind ?month2 (+ (* ?month  100) 10000))
   20  (assert (date (categ today) (month ?month) (day ?day) (dateint (+ ?month2 ?
day2))))
   21 )
   22
   23 (defrule FISH::whatsonthemenu
   24  (fish (species ?sp) (open_season ?open) (close_season ?close)
   25        (habitat $?hb) (seasonal_availability ?sa) (size_min ?min) (limit ?lm)
```

```
   26          (fishing_type ?tp) (fishing_location ?loc) (best_trip $?dn)
(desireability ?ds ))
   27  (date (dateint ?today))
   28  (test ( openseason ?open ?close ?today))
   29   =>
   30   (assert (menu (species ?sp) (habitat $?hb) (seasonal_availability ?sa)
(size_min ?min) (limit ?lm)
   31          (fishing_type ?tp) (fishing_location ?loc) (best_trip $?dn)
(desireability ?ds ) (bonus OFF)))
   32   (printout t crlf ?sp " is open now. They are currently at " ?hb  " and best
fished during the " ?sa  " season. " crlf )
   33   (printout t "We can fish for them by " ?tp " fishing " ?loc ". " crlf   )
   34   (printout t "You can have " ?lm  " fish and it is best to fish for them
during the " $?dn " trip(s). " crlf )
   35   (printout t "On a scale of 1-5 they are desireable: " ?ds crlf crlf )
   36 )
   37
   38 (defrule FISH::testrule
   39  (menu (seasonal_availability $sa3))
   40  =>
   41  (printout t  "==> " $sa3 crlf)
   42 )
   43
   44 (defrule FISH::goodseason
   45  ?f1 <- (menu (species ?sp) (habitat $?hb) (seasonal_availability ?sa)
(size_min ?min) (limit ?lm)
   46         (fishing_type ?tp) (fishing_location ?loc) (best_trip $?dn)
(desireability ?ds )(bonus OFF))
   47  (date (dateint ?today))
   48  (season ( description ?sa )(start_date ?open) (end_date ?close))
   49  (test ( openseason ?open ?close ?today))
   50  =>
   51  (bind ?ds (+ ?ds 2)) ; GOOD SEASON MORE DESIRABLE BY 2
   52  (retract ?f1)
   53  (assert (menu (species ?sp) (habitat $?hb) (seasonal_availability ?sa)
(size_min ?min) (limit ?lm)
   54                          (fishing_type ?tp) (fishing_location ?loc) (best_trip
?dn) (desireability ?ds ) (bonus ON) )
   55  )
   56  (printout t "Oh we are in Season - even better. This is " ?sa " SEASON"
crlf )
   57  (printout t ?sp " is open now. They are currently at " ?hb  " and best
fished NOW!. " crlf )
   58  (printout t "We can fish for them by " ?tp " fishing " ?loc ". " crlf   )
   59  (printout t "You can have " ?lm  " fish and it is best to fish for them
during the " $?dn " trip(s). " crlf )
   60  (printout t "and they are now at desireable level: " ?ds crlf crlf )
   61 )
   62
   63 ;WEATHER MODULE
   64 (focus WEATHER)
   65
   66 (defrule WEATHER::WAVEHEIGHT
```

```
   67    =>
   68    (printout t "What is the expected Wave Heights today =>" crlf )
   69    (bind ?height (read))
   70    (assert (wave_height ( height ?height )))
   71 )
   72
   73 (defrule WEATHER::SWELL
   74    =>
   75    (printout t "What Swell Period in Seconds =>" crlf )
   76    (bind ?sec (read))
   77    (assert (swells( seconds ?sec )))
   78 )
   79
   80 (defrule WEATHER::WIND
   81    =>
   82    (printout t "What is the expected wind speed in knots =>" crlf  )
   83    (bind ?speed (read))
   84    (printout t "From what direction is the wind from? (NORTH SOUTH EAST or
WEST) =>" crlf  )
   85    (bind ?direction (read))
   86    (assert (wind_speed (speed ?speed)(dir ?direction)))
   87 )
   88
   89 (defrule WEATHER::CANCEL
   90       (or (wave_height (height ?ht&:(>= ?ht 10)))
   91          (wind_speed (speed ?sp&:(>= ?sp 30)))
   92          (wind_speed (speed ?sp&:(>= ?sp 20)) (dir EAST))
   93          (swells (seconds ?sec&:(>= ?sec 6)))
   94       )
   95 =>
   96    (assert (acceptable_condition (locale cancel)))
   97    (printout t "Weather is too harsh to fish" crlf )
   98    (exit)
   99 )
  100
  101 (defrule WEATHER::HIGHWINDS
  102    (wind_speed (speed ?sp&:(and (>= ?sp 20 ) (< ?sp 30 ))) (dir ~EAST))
  103  =>
  104    (assert (acceptable_condition (locale inshore) (fsh_type bottom)))
  105 )
  106
  107 (defrule WEATHER::MODERATEWINDS
  108  (wind_speed (speed ?sp&:(and(>= ?sp 15) (< ?sp 20))) (dir WEST))
  109  =>
  110   (assert (acceptable_condition (locale offshore) (fsh_type bottom)))
  111   (assert (acceptable_condition (locale offshore_federal_waters) (fsh_type
bottom)))
  112   (assert (acceptable_condition(locale inshore) (fsh_type bottom)))
  113   (assert (acceptable_condition(locale inshore) (fsh_type game)))
  114   (assert (acceptable_condition(locale inshore) (fsh_type jigging)))
  115   (assert (acceptable_condition(locale inshore) (fsh_type wreck)))
  116  )
  117
```

```
 118 (defrule WEATHER::MODERATE_HARSH_WINDS
 119  (wind_speed (speed ?sp&:(and(>= ?sp 15) (< ?sp 20))) (dir ~WEST))
 120  =>
 121   (assert (acceptable_condition(locale inshore) (fsh_type game)))
 122   (assert (acceptable_condition(locale inshore) (fsh_type bottom)))
 123 )
 124
 125 (defrule WEATHER::CALMSEAS
 126  (wind_speed (speed ?sp&:(< ?sp 15)))
 127  =>
 128   (assert (acceptable_condition (locale offshore_federal_waters) (fsh_type
bottom)))
 129   (assert (acceptable_condition (locale offshore_federal_waters) (fsh_type
game)))
 130   (assert (acceptable_condition (locale offshore_federal_waters) (fsh_type
jigging)))
 131   (assert (acceptable_condition (locale offshore_federal_waters) (fsh_type
wreck)))
 132   (assert (acceptable_condition (locale offshore) (fsh_type bottom)))
 133   (assert (acceptable_condition (locale offshore) (fsh_type game)))
 134   (assert (acceptable_condition (locale offshore) (fsh_type jigging)))
 135   (assert (acceptable_condition (locale offshore) (fsh_type wreck)))
 136   (assert (acceptable_condition (locale inshore) (fsh_type bottom)))
 137   (assert (acceptable_condition (locale inshore) (fsh_type game)))
 138   (assert (acceptable_condition (locale inshore) (fsh_type jigging)))
 139   (assert (acceptable_condition (locale inshore) (fsh_type wreck)))
 140   (assert (acceptable_condition (locale open_ocean) (fsh_type bottom)))
 141   (assert (acceptable_condition (locale open_ocean) (fsh_type game)))
 142   (assert (acceptable_condition (locale open_ocean) (fsh_type jigging)))
 143   (assert (acceptable_condition (locale open_ocean) (fsh_type wreck)))
 144 )
 145
 146 (focus TRIP)
 147 (defrule TRIP::all_possible_trips
 148     ?f1 <- (menu (species ?spec) (habitat $?hab) (size_min ?min)
 149        (limit ?lim) (fishing_type ?ft) (fishing_location ?loc) (best_trip $?)
(desireability ?ds))
 150     ?f2 <- (acceptable_condition (locale ?loc) (fsh_type ?ft))
 151
 152     =>
 153     (retract ?f1)
 154     (assert (all_trips (location ?loc) (method ?ft) (species ?spec)
(desireability ?ds) (check 1)))
 155 )
 156
 157
 158 (defrule TRIP::make_table
 159    ?f1 <- (all_trips (location ?s_loc) (method ?s_ft) )
 160    =>
 161    (assert (table (location ?s_loc) (method ?s_ft) ))
 162 )
 163
 164 (defrule TRIP::eval_function_new
```

```
  165  ?f1 <- (table (location ?s_loc) (method ?s_ft))
  166  ?f2 <- (all_trips (location ?s_loc) (method ?s_ft) (species $?spec)
(desireability ?ds) (check 1))
  167  (not (exists (trips (location ?s_loc ) (method ?s_ft)))) ;restrict to new
entries only
  168  =>
  169  (modify ?f2 (check -1));
  170  (assert (trips (location ?s_loc ) (method ?s_ft) (species ?spec )
(desireability ?ds)))
  171  )
  172
  173 (defrule TRIP::eval_function_cont
  174  ?f3 <- (trips (location ?s_loc ) (method ?s_ft) (species $?species )
(desireability ?sum)) ;current data in trips
  175  ?f1 <- (table (location ?s_loc) (method ?s_ft))
  176  ?f2 <- (all_trips (location ?s_loc) (method ?s_ft) (species $?spec)
(desireability ?ds) (check 1))
  177  =>
  178  (modify ?f2 (check -1))
  179  (retract ?f3)
  180  (assert (trips (location ?s_loc ) (method ?s_ft) (species =(create$ ?species
?spec )) (desireability =(+ ?sum  ?ds)))) ; unify old data with new data
  181  )
  182
  183 (defrule TRIP::send_report
  184    (not (exists (menu (species ?))))
  185    (not (exists(all_trips (check 1 ))));all trips processed
  186    (all_trips (check -1)); and at least one finished
  187    ?f1 <- (print_report (value off))
  188 =>
  189    (modify ?f1 (value on))
  190    (printout t "TURNING ON REPORT" crlf)
  191 )
  192
  193 (focus REPORT)
  194
  195
  196 ( defrule REPORT::MAXIE
  197    ?f2 <- (report_phase header)
  198    ?f1 <- (tops ?)
  199    (trips (location ?s_loc) (method ?s_ft) (species $?species) (desireability
?MAX))
  200    (not (trips (desireability ?LOWER&:(< ?MAX ?LOWER))))
  201 =>
  202    (retract ?f2)
  203    (retract ?f1)
  204    (assert (report_phase choice))
  205    (assert (tops  ?MAX ))
  206    (printout t "MAX PROFIT Points is ==>" ?MAX   crlf crlf crlf crlf)
  207    (printout t "*********Best Trip**********"   crlf )
  208 )
  209
  210 ( defrule REPORT::MAXIE_PRINT
```

```
  211    ?f2<-(report_phase choice)
  212    ?f1<-(tops ?MAX)
  213    (trips (location ?s_loc) (method ?s_ft) (species $?species) (desireability
?MAX))
  214 =>
  215    (retract ?f2)
  216    (assert (report_phase body))
  217    (format t "Trip: %-25s Method: %-6s  Species: %-25s" ?s_loc ?s_ft
(implode$ ?species) )
  218    (printout t crlf "*********Best Trip**********"   crlf crlf crlf
"-------------------------------------------------------------------------"
crlf  )
  219 )
  220
  221 (defrule REPORT::final_report
  222    (report_phase body)
  223    (trips (location ?s_loc) (method ?s_ft) (species $?species) (desireability
?ds))
  224 =>
  225    (format t "Trip: %-25s Method: %-6s  Species: %-25s" ?s_loc ?s_ft
(implode$ ?species) )
  226    (printout t crlf)
  227    (format t "Profit Points: %4d" ?ds)
  228    (printout t crlf crlf  )
  229 )
  230
  231
  232 (get-focus-stack)
  233 (clear-focus-stack)
  234 (reset)
  235 (focus FISH WEATHER TRIP REPORT)
  236 (run)
  237
  238
```

```
  <!--

/* function to open any folds containing a jumped-to line before jumping to it */
function JumpToLine()
{
  var lineNum;
  lineNum = window.location.hash;
  lineNum = lineNum.substr(1); /* strip off '#' */

  if (lineNum.indexOf('L') == -1) {
    lineNum = 'L'+lineNum;
  }
  lineElem = document.getElementById(lineNum);
  /* Always jump to new location even if the line was hidden inside a fold, or
   * we corrected the raw number to a line ID.
   */
  if (lineElem) {
    lineElem.scrollIntoView(true);
  }
  return true;
}
if ('onhashchange' in window) {
  window.onhashchange = JumpToLine;
}

-->  1 (set-dynamic-constraint-checking TRUE)
    2 (defmodule FISH (export deftemplate menu) )
    3 (deftemplate FISH::fish
    4    (slot species )
    5    (multislot habitat )
    6    (slot seasonal_availability ) ;date of fish runs
    7    (slot open_season ) ;the date legal open season begins
    8    (slot close_season ) ;the date the legal season closes
    9    (slot size_min ) ;minimum legal size allowed to be taken
   10    (slot limit ) ;legal maximum allowed number of fish allowed
   11    (slot fishing_type ) ;bottom or gamefishing
   12    (slot fishing_location ) ;inshore or offshore
   13    (multislot best_trip ) ; best to fish day or night
   14    (slot desireability) ; how much people like to fish
   15 )
   16
   17 (deftemplate FISH::menu
   18  (slot species)
   19  (multislot habitat )
   20  (slot seasonal_availability ) ;date of fish runs
   21  (slot size_min ) ;minimum legal size allowed to be taken
   22  (slot limit ) ;legal maximum allowed number of fish allowed
   23  (slot fishing_type ) ;bottom or gamefishing
   24  (slot fishing_location ) ;inshore or offshore
   25  (multislot best_trip ) ; best to fish day or night
   26  (slot desireability) ; how much people like to fish
   27  (slot bonus (type SYMBOL) (allowed-symbols ON OFF) ) ; how much people like
to fish
```

```
   28 )
   29 (deftemplate FISH::season
   30    (slot description ) (slot start_date) (slot end_date)
   31 )
   32 (deftemplate FISH::date
   33     (slot categ (type SYMBOL) (allowed-symbols today start_date end_date) )
   34     (slot day (type INTEGER) (range 1 31))
   35     (slot month (type INTEGER) (range 1 12))
   36     (slot dateint (type INTEGER)(range 20101 21231 )) ; (dateint  - 20000 )
is the date
   37 )
   38
   39 (deffunction FISH::openseason (?start ?end ?today )
   40    (if (> ?start ?end )
   41     then
   42      (or
   43          (and (>= ?today ?start ) (<= ?today  21231))
   44          (and (>= ?today 20101 ) (<= ?today ?end))
   45       )
   46      else (and (>= ?today ?start) (<= ?today ?end))
   47    )
   48 )
   49
   50 (deffacts FISH::fishing_seasons
   51    ( season (description spring_bass_run) (start_date  20401) (end_date
20515) )
   52    ( season (description fall_bass_run) (start_date 21001) (end_date
21201) )
   53    ( season (description winter) (start_date  21215) (end_date 20315) )
   54    ( season (description winter_tautog) (start_date  21015) (end_date
20315) )
   55    ( season (description spring_tautog ) (start_date 20401) (end_date
20515) )
   56    ( season (description warm_months ) (start_date 20415) (end_date 21101) )
   57 )
   58
   59
   60 (deffacts FISH::fishes
   61    (fish (species tautog) (habitat bottom wrecks) (seasonal_availability
winter_tautog )
   62          (open_season 21022) (close_season 21231 ) (size_min 24 )
   63          (limit 4 ) (fishing_type bottom ) (fishing_location offshore )
   64          (best_trip day ) (desireability 5)
   65    )
   66    (fish (species tautog) (habitat bottom wrecks) (seasonal_availability
winter_tautog )
   67          (open_season 21022) (close_season 21231 ) (size_min 24 )
   68          (limit 4 ) (fishing_type bottom ) (fishing_location
offshore_federal_waters )
   69          (best_trip day ) (desireability 5)
   70    )
   71    (fish (species tautog) (habitat bottom wrecks) (seasonal_availability
winter_tautog )
```

```
   72          (open_season 21022) (close_season 21231 ) (size_min 24 )
   73          (limit 4 ) (fishing_type bottom ) (fishing_location inshore )
   74          (best_trip day ) (desireability 2)
   75    )
   76    (fish (species striped_bass) (habitat inshore current rocks )
(seasonal_availability spring_bass_run )
   77          (open_season 20101 ) (close_season 21231 ) (size_min 31  )
   78          (limit 1 ) (fishing_type game) (fishing_location inshore )
   79          (best_trip day night ) (desireability 4)
   80    )
   81    (fish (species striped_bass) (habitat inshore current rocks )
(seasonal_availability fall_bass_run )
   82          (open_season 20101) (close_season 21231 ) (size_min 31 )
   83          (limit 1 ) (fishing_type game ) (fishing_location inshore )
   84          (best_trip day night ) (desireability 4)
   85    )
   86    (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
   87          (open_season 20501 ) (close_season 20831 ) (size_min 10 )
   88          (limit 30 ) (fishing_type bottom ) (fishing_location inshore )
   89          (best_trip day ) (desireability 3 )
   90    )
   91    (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
   92          (open_season 20501 ) (close_season 20831 ) (size_min 10 )
   93          (limit 30 ) (fishing_type bottom ) (fishing_location offshore )
   94          (best_trip day ) (desireability 4 )
   95    )
   96    (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
   97          (open_season 20501 ) (close_season 20831 ) (size_min 10 )
   98          (limit 30 ) (fishing_type bottom ) (fishing_location
offshore_federal_waters )
   99          (best_trip day ) (desireability 4 )
  100    )
  101    (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
  102          (open_season 20901 ) (close_season 21031 ) (size_min 10 )
  103          (limit 45 ) (fishing_type bottom ) (fishing_location offshore )
  104          (best_trip day night) (desireability 4 )
  105    )
  106    (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
  107          (open_season 20901 ) (close_season 21031 ) (size_min 10 )
  108          (limit 45 ) (fishing_type bottom ) (fishing_location
offshore_federal_waters )
  109          (best_trip day night) (desireability 4 )
  110    )
  111    (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
  112          (open_season 20901 ) (close_season 21031 ) (size_min 10 )
  113          (limit 45 ) (fishing_type bottom ) (fishing_location inshore )
  114          (best_trip day night) (desireability 3 )
  115    )
  116    (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
  117          (open_season 21101 ) (close_season 21231 ) (size_min 10 )
  118          (limit 30 ) (fishing_type bottom ) (fishing_location inshore )
  119          (best_trip day ) (desireability 3 )
  120    )
```

```
  121    (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
  122          (open_season 21101 ) (close_season 21231 ) (size_min 10 )
  123          (limit 30 ) (fishing_type bottom ) (fishing_location offshore )
  124          (best_trip day ) (desireability 4 )
  125    )
  126    (fish (species scup) (habitat bottom) (seasonal_availability warm_months )
  127          (open_season 21101 ) (close_season 21231 ) (size_min 10 )
  128          (limit 30 ) (fishing_type bottom ) (fishing_location
offshore_federal_waters )
  129          (best_trip day ) (desireability 4 )
  130    )
  131    (fish (species fluke ) (habitat bottom ) (seasonal_availability
warm_months)
  132          (open_season 20517) (close_season 20921 ) (size_min 18 )
  133          (limit 5 ) (fishing_type bottom ) (fishing_location inshore )
  134          (best_trip day ) (desireability 3)
  135    )
  136    (fish (species bluefish) (habitat open_water current )
(seasonal_availability all_year)
  137          (open_season 20101 ) (close_season 21231 ) (size_min 0 )
  138          (limit 500 ) (fishing_type game ) (fishing_location inshore )
  139          (best_trip day ) (desireability 2 )
  140    )
  141    (fish (species bluefish) (habitat open_water current )
(seasonal_availability all_year)
  142          (open_season 20101 ) (close_season 21231 ) (size_min 12 )
  143          (limit 15 ) (fishing_type game ) (fishing_location offshore )
  144          (best_trip day ) (desireability 3 )
  145    )
  146    (fish (species bluefish) (habitat open_water current )
(seasonal_availability all_year)
  147          (open_season 20101 ) (close_season 21231 ) (size_min 12 )
  148          (limit 15 ) (fishing_type game ) (fishing_location
offshore_federal_waters )
  149          (best_trip day ) (desireability 3 )
  150    )
  151    (fish (species cod ) (habitat open_water) (seasonal_availability winter )
  152          (open_season 20101 ) (close_season 21231 ) (size_min 22 )
  153          (limit 10 ) (fishing_type bottom ) (fishing_location offshore )
  154          (best_trip day ) (desireability 3)
  155    )
  156    (fish (species cod ) (habitat open_water) (seasonal_availability winter )
  157          (open_season 20101 ) (close_season 21231 ) (size_min 22 )
  158          (limit 10 ) (fishing_type bottom ) (fishing_location
offshore_federal_waters )
  159          (best_trip day ) (desireability 3)
  160    )
  161    (fish (species black_sea_bass ) (habitat bottom ) (seasonal_availability
warm_months )
  162          (open_season 20715) (close_season 21031 ) (size_min 14 )
  163          (limit 8 ) (fishing_type bottom ) (fishing_location inshore )
  164          (best_trip day ) (desireability 3 )
  165    )
```

```
  166    (fish (species black_sea_bass ) (habitat bottom ) (seasonal_availability
warm_months )
  167          (open_season 21101) (close_season 21231 ) (size_min 14 )
  168          (limit 10 ) (fishing_type bottom ) (fishing_location inshore )
  169          (best_trip day ) (desireability 3 )
  170    )
  171    (fish (species black_sea_bass ) (habitat bottom ) (seasonal_availability
warm_months )
  172          (open_season 20715) (close_season 21031 ) (size_min 14 )
  173          (limit 8 ) (fishing_type bottom ) (fishing_location offshore )
  174          (best_trip day ) (desireability 3 )
  175    )
  176    (fish (species black_sea_bass ) (habitat bottom ) (seasonal_availability
warm_months )
  177          (open_season 21101) (close_season 21231 ) (size_min 14 )
  178          (limit 10 ) (fishing_type bottom ) (fishing_location offshore )
  179          (best_trip day ) (desireability 3 )
  180    )
  181    (fish (species black_sea_bass_federal ) (habitat bottom )
(seasonal_availability warm_months )
  182          (open_season 20515) (close_season 20921 ) (size_min 14 )
  183          (limit 10 ) (fishing_type bottom ) (fishing_location
offshore_federal_waters )
  184          (best_trip day ) (desireability 3 )
  185    )
  186    (fish (species black_sea_bass_federal ) (habitat bottom )
(seasonal_availability warm_months )
  187          (open_season 21022) (close_season 21231 ) (size_min 14 )
  188          (limit 10 ) (fishing_type bottom ) (fishing_location
offshore_federal_waters )
  189          (best_trip day ) (desireability 3 )
  190    )
  191    (fish (species ling ) (habitat mud bottom ) (seasonal_availability
winter )
  192          (open_season 20101) (close_season 21231 ) (size_min  0)
  193          (limit 500 ) (fishing_type bottom ) (fishing_location offshore )
  194          (best_trip day ) (desireability 2)
  195    )
  196    (fish (species ling ) (habitat mud bottom ) (seasonal_availability
winter )
  197          (open_season 20101) (close_season 21231 ) (size_min  0)
  198          (limit 500 ) (fishing_type bottom ) (fishing_location
offshore_federal_waters )
  199          (best_trip day ) (desireability 2)
  200    )
  201 )
  202
```

```
  <!--

/* function to open any folds containing a jumped-to line before jumping to it */
function JumpToLine()
{
  var lineNum;
  lineNum = window.location.hash;
  lineNum = lineNum.substr(1); /* strip off '#' */

  if (lineNum.indexOf('L') == -1) {
    lineNum = 'L'+lineNum;
  }
  lineElem = document.getElementById(lineNum);
  /* Always jump to new location even if the line was hidden inside a fold, or
   * we corrected the raw number to a line ID.
   */
  if (lineElem) {
    lineElem.scrollIntoView(true);
  }
  return true;
}
if ('onhashchange' in window) {
  window.onhashchange = JumpToLine;
}

--> 1 (defmodule WEATHER (import FISH deftemplate menu) (export deftemplate
acceptable_condition))
   2 (deftemplate WEATHER::acceptable_condition ;these are viable non-species
specific fishing options based on weather
   3    (slot locale  (type SYMBOL )(allowed-symbols cancel inshore offshore
open_ocean offshore_federal_waters) )
   4    (slot fsh_type  (type SYMBOL )(allowed-symbols bottom game jigging wreck) )
   5 )
   6
   7 (deftemplate WEATHER::wave_height
   8    (slot height (type NUMBER ))
   9 )
  10 (deftemplate WEATHER::swells
  11    (slot seconds (type NUMBER ))
  12 )
  13 (deftemplate WEATHER::wind_speed
  14    (slot speed (type NUMBER ))
  15    (slot dir (type SYMBOL )(allowed-symbols EAST WEST NORTH SOUTH ))
  16 )
  17
```

```
  <!--

/* function to open any folds containing a jumped-to line before jumping to it */
function JumpToLine()
{
  var lineNum;
  lineNum = window.location.hash;
  lineNum = lineNum.substr(1); /* strip off '#' */

  if (lineNum.indexOf('L') == -1) {
    lineNum = 'L'+lineNum;
  }
  lineElem = document.getElementById(lineNum);
  /* Always jump to new location even if the line was hidden inside a fold, or
   * we corrected the raw number to a line ID.
   */
  if (lineElem) {
    lineElem.scrollIntoView(true);
  }
  return true;
}
if ('onhashchange' in window) {
  window.onhashchange = JumpToLine;
}

--> 1 (defmodule TRIP (export deftemplate trips) (import FISH deftemplate menu)
(import WEATHER deftemplate acceptable_condition))
   2
   3 (deftemplate TRIP::all_trips
   4     (slot location ) (slot method) (multislot species) (slot desireability)
(slot check (type INTEGER))
   5 )
   6 (deftemplate TRIP::table
   7    (slot location) (slot method)
   8    )
   9
  10 (deftemplate TRIP::trips
  11     (slot location ) (slot method) (multislot species) (slot desireability)
  12 )
  13
  14 (deftemplate TRIP::spec_tmp (multislot spec) )
  15 (deftemplate TRIP::print_report (slot value) )
  16 (deffacts TRIP::inital
  17    (spec_tmp)
  18    (sum 0)
  19    (print_report (value off))
  20 )
  21
```

```
  <!--

/* function to open any folds containing a jumped-to line before jumping to it */
function JumpToLine()
{
  var lineNum;
  lineNum = window.location.hash;
  lineNum = lineNum.substr(1); /* strip off '#' */

  if (lineNum.indexOf('L') == -1) {
    lineNum = 'L'+lineNum;
  }
  lineElem = document.getElementById(lineNum);
  /* Always jump to new location even if the line was hidden inside a fold, or
   * we corrected the raw number to a line ID.
   */
  if (lineElem) {
    lineElem.scrollIntoView(true);
  }
  return true;
}
if ('onhashchange' in window) {
  window.onhashchange = JumpToLine;
}

-->1 (defmodule REPORT (import TRIP deftemplate trips))
  2 (deffacts REPORT::initial_facts
  3    (report_phase header)
  4    (tops 0)
  5 )
  6
```