Logic Programming (PLP 11)
Prolog Imperative Control Flow:
Backtracking, Cut, Fail, Not

Carlos Varela
Rensselaer Polytechnic Institute

February 1, 2010
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).

_C = _X

success
cold(seattle) fails;
backtrack.

cold(rochester)

X = rochester

OR

AND

X = seattle

OR

rainy(rochester)
rainy(seattle)
Imperative Control Flow

- Programmer has *explicit control* on backtracking process.

Cut (!)

- As a goal it succeeds, but with a *side effect*:
  - Commits interpreter to choices made since unifying parent goal with left-hand side of current rule.
Cut (!) Example

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

\_C = \_X

\text{GOAL FAILS.}
Cut (!) Example 2

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

C = troy FAILS
snowy(X) is committed to bindings (X = seattle).
GOAL FAILS.

C = troy

snowy(C)

OR

snowy(troy)

cold(rochester)

cold(X)

AND

rainy(X)

_X = X

X = seattle

rainy(seattle)

rainy(rochester)

OR

FAILS

snowy(X) is committed to bindings (X = seattle).
GOAL FAILS.
Cut (!) Example 3

rainy(seattle) :- !.
rainy(rochester).
cold(rochester).
snowy(X) :- rainy(X), cold(X).
snowy(troy).
Cut (!) Example 3

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

\[ \text{C = troy SUCCEEDS} \]
Only rainy(X) is committed to bindings (X = seattle).

\[ _C = _X \]
\[ X = \text{seattle} \]
\[ \text{C. Varela} \]
Cut (!) Example 4

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

Cut (!) Example 4

_C = _X

X = seattle

rainy(seattle)

rainy(X)

OR

OR

cold(X)

_ = _C

snowy(X)
snowy(C)

AND

success

cold(seattle) fails; backtrack.

cold(rochester)

X = rochester

rainy(rochester)

cold(rochester)
Cut (!) Example 5

rainy(seattle).
rainy(rochester).
cold(rochester).
snowy(X) :- rainy(X), cold(X), !.
Cut (!) Example 5

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

\[
\begin{align*}
\text{snowy}(X) & \quad \text{AND} \\
\text{rainy}(X) & \quad \text{OR} \\
\text{cold}(X) & \\
\end{align*}
\]

\[
\begin{align*}
_X & = X \\
_C & = C \\
\text{success} & \\
\end{align*}
\]
# First-Class Terms

<table>
<thead>
<tr>
<th>Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>call(P)</code></td>
<td>Invoke predicate as a goal.</td>
</tr>
<tr>
<td><code>assert(P)</code></td>
<td>Adds predicate to database.</td>
</tr>
<tr>
<td><code>retract(P)</code></td>
<td>Removes predicate from database.</td>
</tr>
<tr>
<td><code>functor(T,F,A)</code></td>
<td>Succeeds if $T$ is a term with $F$ and $A$.</td>
</tr>
</tbody>
</table>

Where $T$ is a term with functor $F$ and arity $A$. 

\[ \text{not } P \text{ is not } \neg P \]

- In Prolog, the database of facts and rules includes a list of things assumed to be true.

- It does not include anything assumed to be false.

- Unless our database contains everything that is true (the closed-world assumption), the goal not P (or \(+ P\) in some Prolog implementations) can succeed simply because our current knowledge is insufficient to prove P.
More not vs ¬

?- snowy(X).
X = rochester
?- not(snowy(X)).
no

Prolog does not reply: \texttt{X = seattle}.

The meaning of \texttt{not(snowy(X))} is:

\[ \neg \exists x \ [ \text{snowy}(X) ] \]

rather than:

\[ \exists x \ [ \neg \text{snowy}(X) ] \]
Fail, true, repeat

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fail</td>
<td>Fails current goal.</td>
</tr>
<tr>
<td>true</td>
<td>Always succeeds.</td>
</tr>
<tr>
<td>repeat</td>
<td>Always succeeds, provides infinite choice points.</td>
</tr>
</tbody>
</table>

repeat.
repeat :- repeat.
not Semantics

not(P) :- call(P), !, fail.
not(P).

Definition of \texttt{not} in terms of failure (\texttt{fail}) means that variable bindings are lost whenever \texttt{not} succeeds, e.g.:

\begin{verbatim}
?- not(not(snowy(X))).
X=_G147
\end{verbatim}
Conditionals and Loops

statement :- condition, !, then.
statement :- else.

natural(1).
natural(N) :- natural(M), N is M+1.
my_loop(N) :- N>0,
        natural(I), I<=N,
        write(I), nl,
        I=N,
        !, fail.

Also called generate-and-test.

C. Varela
Prolog lists

- \[a, b, c\] is syntactic sugar for:

\[(a, .(b, .(c, []))))\]

where \[[]\] is the empty list, and \[.\] is a built-in cons-like functor.

- \[a, b, c\] can also be expressed as:

\[a \mid [b, c]\quad\text{or}\quad[a, b \mid [c]]\quad\text{or}\quad[a, b, c \mid []]\]
Prolog lists append example

append([], L, L).
append([H|T], A, [H|L]) :- append(T, A, L).
8. What do the following Prolog queries do?

?- repeat.

?- repeat, true.

?- repeat, fail.

Corroborate your thinking with a Prolog interpreter.

9. Draw the search tree for the query “\texttt{not(not(snowy(City)))}”. When are variables bound/unbound in the search/backtracking process?

10. PLP Exercise 11.6 (pg 571).

11. PLP Exercise 11.7 (pg 571).