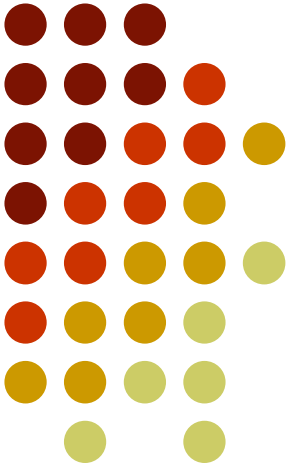
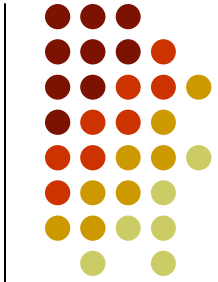
Introduction to Prolog
read, write, assert, retract

CS171: Expert Systems

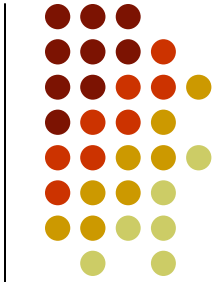


Topics:



- Read and write predicates
- Assert and retract predicates
- Cuts
- Tracing

Write predicate

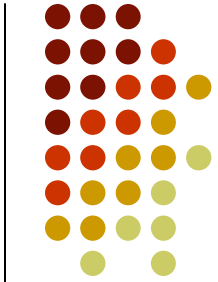


- write( ) Writes a single term to the terminal.
- For example: write(a), or write('How are you?')

- write_ln() Writes a term to the terminal followed by a new line.

- tab(X) Writes an X number of spaces to the terminal.

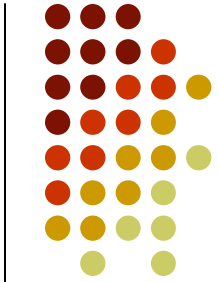
Read predicate



- read(X) Reads a term from the keyboard and instantiates variable X to the value of the read term.
- This term to be read has to be followed by a dot "." and a white space character (such as an enter or space).
- For example:

```
hello :-
        write('What is your name ?'),
        read(X),
        write('Hello'), tab(1), write(X).
```

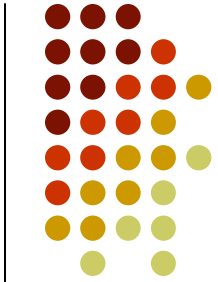
Assert predicate



- assert(X) — Adds a new fact or clause to the database. Term is asserted as the last fact or clause with the same key predicate.
- asserta(X) — Same as assert, but adds a clause at the beginning of the database
- assertz(X) — Exactly the same as **assert(X)**

- ‘a’ being the first letter and ‘z’ being the last letter of the alphabet should remind you where in the database you are adding a new fact or a clause.

Assert predicate

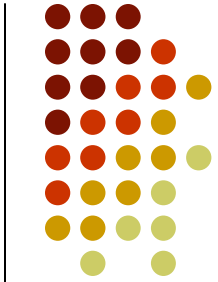


- For example:

```
:- dynamic good/2.
:- dynamic bad/2.
assert(good(skywalker, luke)).
assert(good(solo, han)).
assert(bad(vader, darth)).

?- listing(good).
```

Retract predicate



- retract(X) Removes fact or clause X from the database.
- retractall(X) Removes all facts or clauses from the database for which the head unifies with X.
- For example:

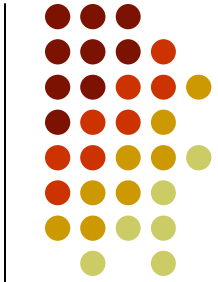
  retract(bad(vader, darth)).

  retractall(good(_, _)).

  ?- good(X, Y).

  No

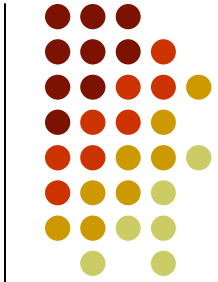
Cuts



- ! is a Prolog feature called the cut.
- Cuts may be inserted anywhere within a clause to prevent backtracking to previous subgoals.
- For example:

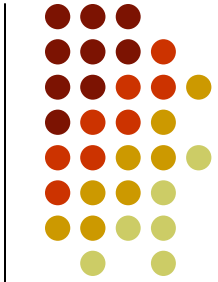
  a(X) :- b(X), c(X), !, d(X), e(X).

Cuts



- What do cuts do?
- Suppose that clause a() from the previous slide has been invoked and the subgoals b(X) and c(X) have been satisfied. On encountering the cut:
  - The cut will succeed and Prolog will try to satisfy subgoals d(X) and e(X).
  - If d(X) and e(X) succeed then a(X) succeeds.
  - If d(X) and e(X) do not succeed and backtracking returns to the cut, then the backtracking process will immediately terminate and a(X) fails.

Cuts



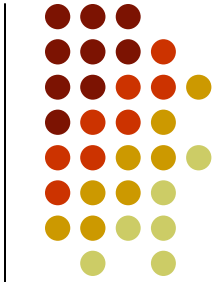
- For example:

```
max(A,B,B) :- A < B.
max(A,B,A).
?- max(3,4,M).
M = 4   ;
M = 3
```

- Using a cut:

```
max(A,B,B) :- A < B, !.
max(A,B,A).
?- max(3,4,M).
M = 4 ;
No
```

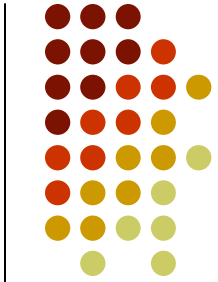
Cuts



- Cuts are commonly used in the generate-and-test programming paradigm:

```
find_just_one_solution(X) :-
        candidate_solution(X),
        test_solution(X),
        !.
```

Cuts

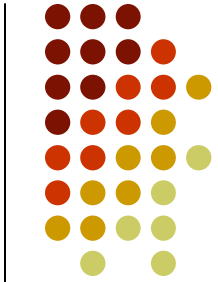


- cut – fail combination cab be used to express negation:
- For example:

```
nonsibling(X, Y) :- sibling(X, Y), !, fail.
nonsibling(X, Y).
```

- Fail — A predicate that always returns false (i.e. it always fails, causing the whole clause to fail and Prolog to try another branch in the recursion tree.

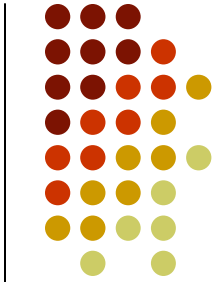
Trace predicate



- The trace predicate prints out information about the sequence of goals in order to show where the program has reached in its execution.

- Example (see trace_example.pl on the course web site)

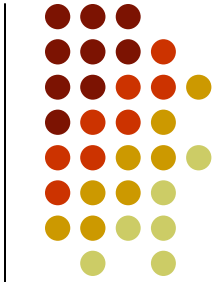
Trace predicate



- Some of the events which may happen during a trace:

- CALL: A CALL event occurs when Prolog tries to satisfy a goal
- EXIT: An EXIT event occurs when some goal has just been satisfied
- REDO: A REDO event occurs when the system comes back to a goal, trying o re-satisfy it
- FAIL: A FAIL event occurs when a goal fails

Reference



- Clocksin, W.F., and Mellish C.S. *Programming in Prolog*. 4th edition. New York: Springer-Verlag. 1994.
- Van Le, T. *Techniques of Prolog Programming*. John Wiley & Sons, Inc. 1993.