KIIS: Artificial Intelligence and Artificial Intelligence Lecture 6/13 sep 2005 Henning Christiansen

Self-inspecting and self-modifying programs

- Tools in Prolog
- Meta-interpreters (short intro)
- Modifying program while it runs

Metaprogramming: treating programs as data

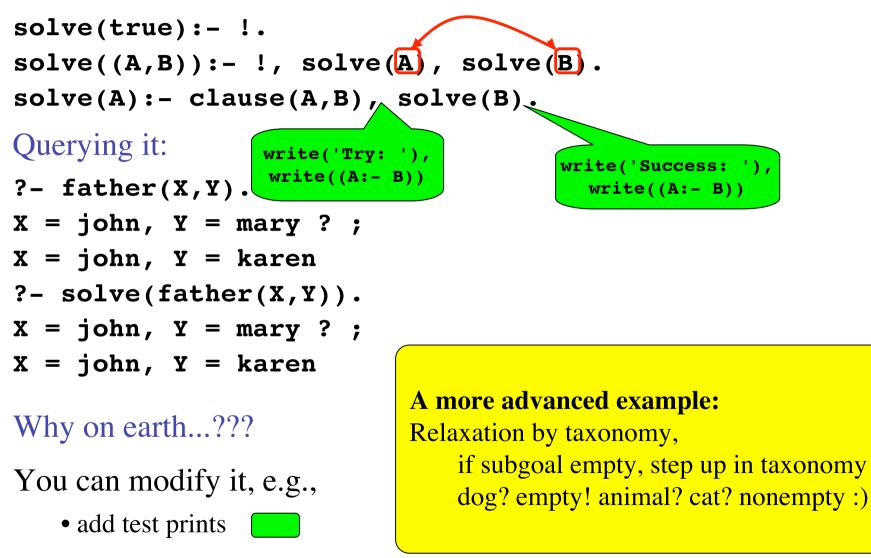
Homogeneous format, program \approx data

 Can you tell difference between fact p(a,b) and term p(a,b)?
 clause p(x):- q(x) and term p(x):- q(x)?

Self-inspection by predicate clause/2.
Works as if any clause head:- body represented dually by fact
clause(head, body).

- NB: works only when predicates are declared to be dynamic:
 - :- dynamic father/2, grandfather/2.

Vanilla: A (meta-)interpreter for Prolog in Prolog



change order of evaluation ~

Modifying the program while it runs Add new *last* clause: assertz(*clause*) Add new *first* clause: asserta(*clause*) Delete first clause unifying pattern: retract(clause) ?- father(X,Y). X = peter, Y = karen ?; // no?- asserta(father(john,mary)),assertz(father(john,paul)). yes ?- father(X,Y).X = john, Y = mary ?;X = peter, Y = karen ?;X = john, Y = paul ?; // no?- \+ (retract(father(john,X)), fail). // yes ?- father(X,Y). X = peter, Y = karen ?; // no

Applications for AI

- Metainterpreters, for modifying execution strategy, adding "meta-rules" (a la expert systems)
- Defining backward/forward chaining with assert for "modifying the fact base"
 - (I have never tried this; a good exercise ...)
- Later we introduce Constraint Handling Rules in which these things can be explained in more clean way.