CS 420 Advanced Programming Languages
Fall Semester, 2022
Doc 23 Prolog 4
Nov 15, 2022

Some Tools

Profile
Debug
Threads
Foreign Language Interface
Standalone Executables

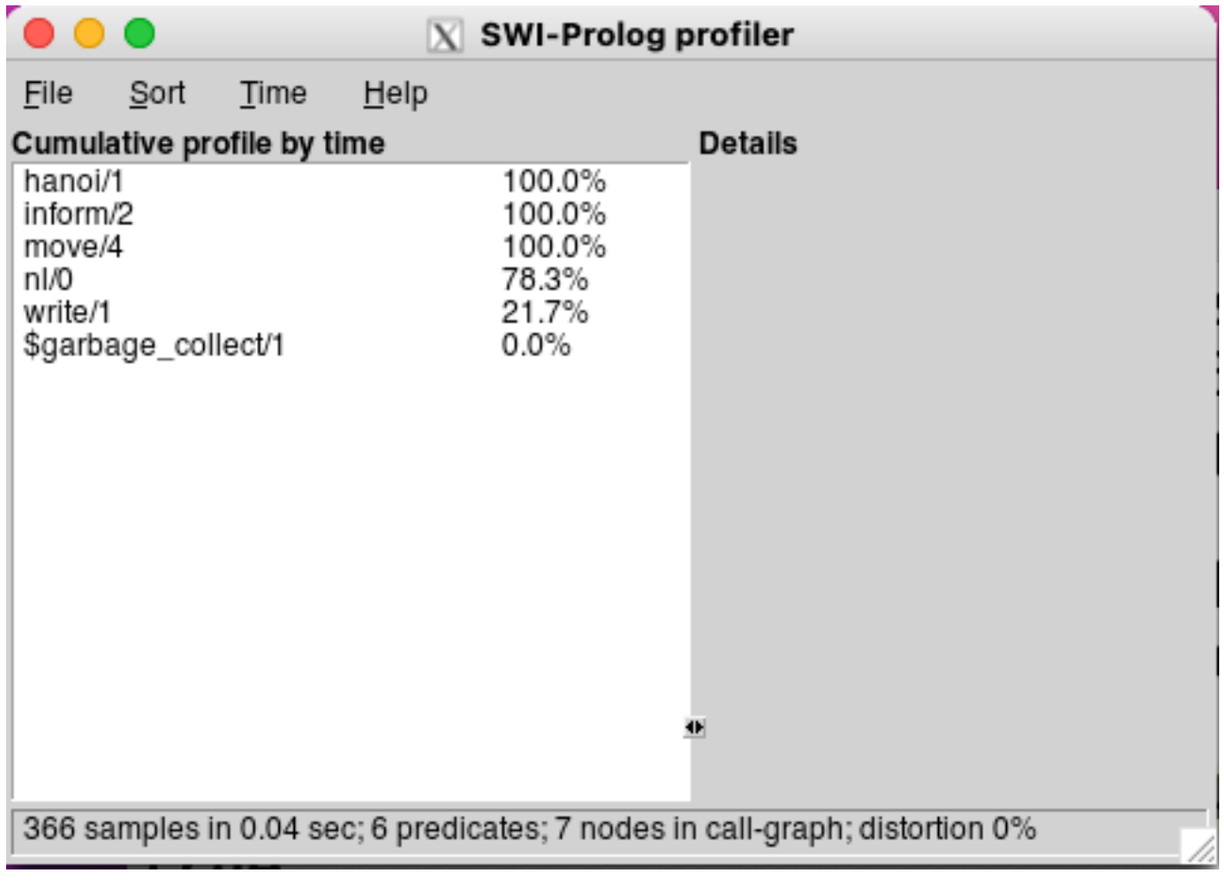
Profiling

```
hanoi(N) :- move(N,left, center, right).

move(0,_,_,_) :- !.
move(N, A, B,C) :-
        M is N -1,
        move(M, A, C, B), inform(A, B), move(M,C,B,A).

inform(X, Y) :-
        write([move,from,X,to,Y]),
        nl.
```

```
?- profile(hanoi(15)).
```



Debugging

?- gspy(hanoi).

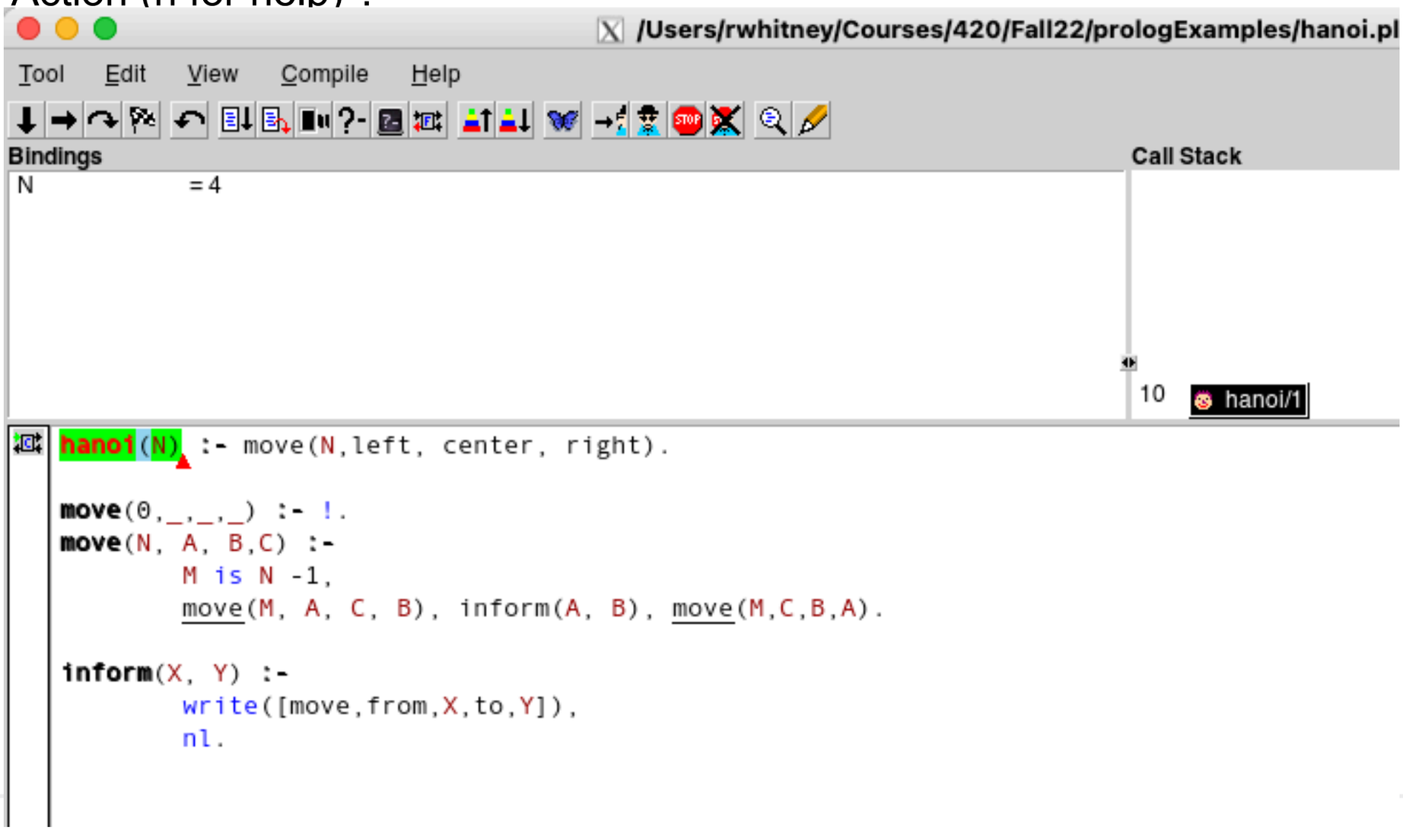
% The graphical front-end will be used for subsequent tracing

% Spy point on hanoi/1

true.

[debug] ?- hanoi(4).

Action (h for help) ?



Threads

?- thread_create(hanoi(10),ID,[]).

All internal Prolog operations are thread-safe

mutex

mutex_create(?MutexId)
mutex_lock(+MutexId)
mutex_unlock(+MutexId)
mutex_unlock_all

Message Queues

thread_send_message(+QueueOrThreadId, +Term)
thread_get_message(?Term)
message_queue_create(?Queue)

Foreign Language Interface

SWI Prolog has an interface to C

C can call Prolog predicates

Standalone Executables

qsave_program(File)

qsave_program(File, [options)

Can bundle facts and rules into one file

Remove K'th Element

remove_at(RemovedElement,L,K,ResultantList)
  Remove the K'th element from list L,
  RemovedElement is the removed element
  ResultantList is L with element removed

```
remove_at(X,[X|Xs],1,Xs).
remove_at(X,[Y|Xs],K,[Y|Ys]) :- K > 1,
K1 is K - 1, remove_at(X,Xs,K1,Ys).
```

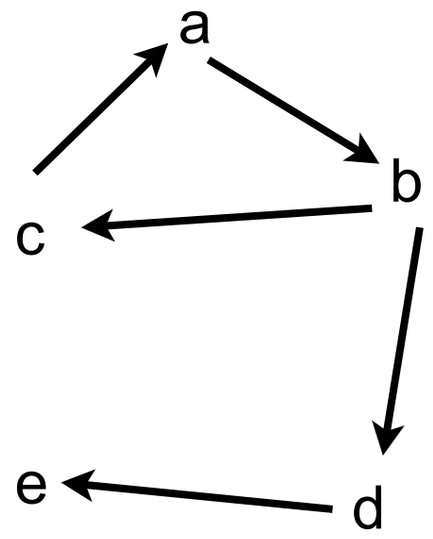
insert

insert_at(Element,L,K,ResultantList)
Put Element at location K in list L, ResultantList is the result

```
insert_at(X,L,K,R) :- remove_at(X,R,K,L).
```

Graphs

```
edge(a,b).
edge(b,c).
edge(c,a).
edge(b,d).
edge(d,e).
```




```
graph([a,b,c,d,e],[e(a,b),e(b,c),e(c,a), e(b,d),e(d,e)]).
```

```
[n(a,[b]), n(b,[c,d]), n(c,[a]), n(d,[e]), n(e, [])]
```

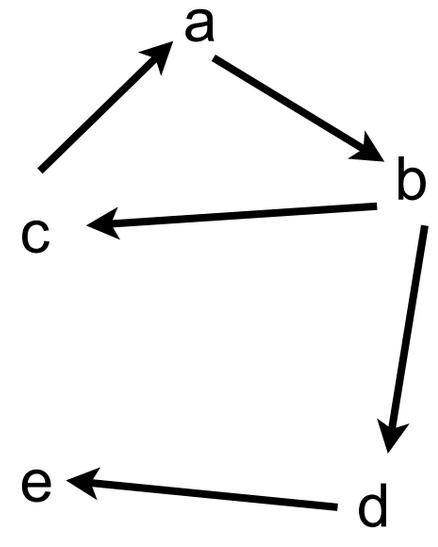
```
[a-b, b-c, c-a, b-d, d-e]
```

Search

```
edge(a,b).
edge(b,c).
edge(c,a).
edge(b,d).
edge(d,e).

go(X,Y) :- go(X,Y,[]), !.
go(X,X,T).
go(X,Y,T) :- edge(X,Z), legal(Z,T),go(Z,Y, [Z|T]).

legal(X,[]).
legal(X, [H|T]) :- \+X = H, legal(X,T).
```

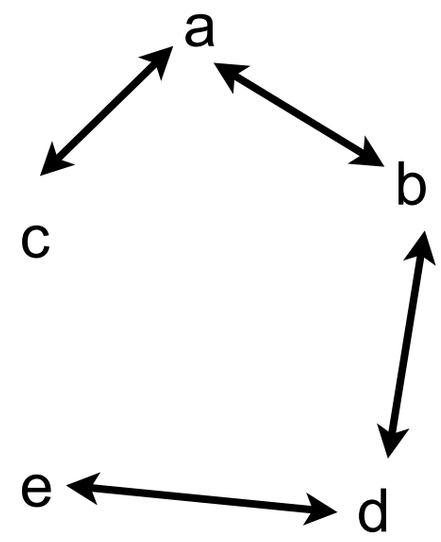


Search

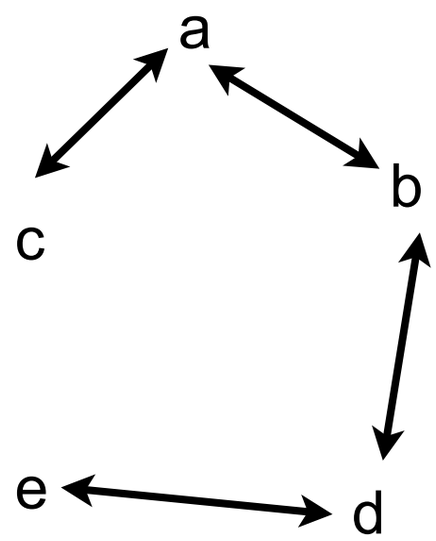
```
edge(a,b).
edge(b,c).
edge(c,a).
edge(b,d).
edge(d,e).

go(X,Y) :- go(X,Y,[]), !.
go(X,X,T).
go(X,Y,T) :-
   (edge(X,Z);edge(Z,X)),
   legal(Z,T),
   go(Z,Y, [Z|T]).

legal(X,[]).
legal(X, [H|T]) :- \+X = H, legal(X,T).
```



With Route



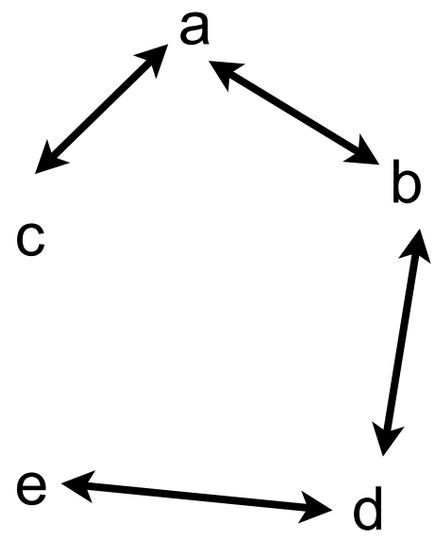

```
edge(a,b).
edge(b,c).
edge(c,a).
edge(b,d).
edge(d,e).

go(Start,Destination,Route) :- go(Start,Destination,[],R ), reverse(R, Route).
go(X,X,T, [X|T]).
go(Place,Y,T,R) :-
   legal_node(Place,T, Next),
   go(Next,Y,[Place|T], R).

legal_node(X, Trail, Y):- (edge(X,Y);edge(Y,X)), legal(Y, Trail).

legal(X,[]).
legal(X, [H|T]) :- \+X = H, legal(X,T).
```

With Explicit Routes




```
edge(a,b).
edge(b,c).
edge(c,a).
edge(b,d).
edge(d,e).
```

```
go(Start,Destination,Route) :- go1([[Start]],Destination,R ), reverse(R, Route).

go1([First|Rest],Destination, First) :- First = [Destination|_].
go1([[Last|Trail]|Others],Destination,Route) :-
    findall([Z,Last|Trail],legal_node(Last,Trail,Z),List),
    append(List,Others, NewRoutes),
    go1(NewRoutes,Destination, Route).

legal_node(X, Trail, Y):- (edge(X,Y);edge(Y,X)), legal(Y, Trail).

legal(X,[]).
legal(X, [H|T]) :- \+X = H, legal(X,T).
```