

Introduction to Prolog (2)

Using some slides from Peter Flach's materials

Recap from the last lecture... Herbrand...

- Herbrand universe, Herbrand base, Herbrand interpretation, Herbrand model

Recap from the last lecture... Hebrand...

- Herbrand universe, Herbrand base, Herbrand interpretation, Herbrand model
- For instance:

```
teaches(alice, prolog).  
course(X) :- teaches(_,X).  
person(X) :- teaches(X,_).
```

Recap from the last lecture... Hebrand...

- Herbrand universe, Herbrand base, Herbrand interpretation, Herbrand model

- For instance:

```
teaches(alice, prolog).  
course(X) :- teaches(_,X).  
person(X) :- teaches(X,_).
```

- Hebrand universe:

- {alice, prolog}

Recap from the last lecture... Herbrand...

- Herbrand universe, Herbrand base, Herbrand interpretation, Herbrand model

- For instance:

```
teaches(alice, prolog).  
course(X) :- teaches(_,X).  
person(X) :- teaches(X,_).
```

- Herbrand universe:

- {alice, prolog}

- Herbrand base:

- {teaches(alice, prolog), teaches(prolog,alice), teaches(prolog,prolog), teaches(alice,alice),
course(alice), course(prolog), person(alice), person(prolog)}

Recap from the last lecture... Hebrand...

- Herbrand universe, Herbrand base, Herbrand interpretation, Herbrand model

- For instance:

```
teaches(alice, prolog).  
course(X) :- teaches(_,X).  
person(X) :- teaches(X,_).
```

- **Herbrand universe:**

- {alice, prolog}

- **Herbrand base:**

- {teaches(alice, prolog), teaches(prolog,alice), teaches(prolog,prolog), teaches(alice,alice), course(alice), course(prolog), person(alice), person(prolog)}

- **Herbrand interpretation:**

- {teaches(alice, prolog), course(prolog), person(alice)}
- ...which is also a Herbrand model of the program.

Recap from the last lecture... Functions

- **Function symbols** allow us to construct composite terms (besides that they also allow us to do **Skolemization**).

Recap from the last lecture... Functions

- **Function symbols** allow us to construct composite terms (besides that they also allow us to do **Skolemization**).

- **Example:**

`num(0) .`

`num(s(X)) :- num(X) .`

Recap from the last lecture... Functions

- **Function symbols** allow us to construct composite terms (besides that they also allow us to do **Skolemization**).

- **Example:**

$\text{num}(0) .$

$\text{num}(s(X)) \text{ :- num}(X) .$

- **Herbrand universe:** $\{0, s(0), s(s(0)), \dots\}$
- **Hebrand base:** $\{\text{num}(0), \text{num}(s(0)), \text{num}(s(s(0))), \dots\}$
- In this case, we do not have a finite Herbrand model.

A Digression: Skolemization

“Everybody knows somebody.”

A Digression: Skolemization

“Everybody knows somebody.”

Skolemization to avoid an existential quantifier

```
knows (X, person_known_by (X) ) .
```

functor

term

complex term

```
knows (peter, person_known_by (peter) ) .
```

```
knows (anna, person_known_by (anna) ) .
```

```
knows (paul, person_known_by (paul) ) .
```

...

Recap from the last lecture... Logical consequence

- A clause C is a **logical consequence** of a program (set of clauses) P iff every model of P is a model of C .

Recap from the last lecture... Logical consequence

- A clause **C** is a **logical consequence** of a program (set of clauses) **P** iff every model of **P** is a model of **C**.

Program **P**:

```
num(0) .  
num(s(X)) :- num(X) .
```

Recap from the last lecture... Logical consequence

- A clause **C** is a **logical consequence** of a program (set of clauses) **P** iff every model of **P** is a model of **C**.

Program **P**:

```
num(0) .  
num(s(X)) :- num(X) .
```

Some logical consequences of **P**:

```
num(s(s(0))) .  
num(0)  
num(s(s(X))) :- num(X) .
```

An Example: Prague Metro



A Prolog DB (1)

```
connected(nemocnice_motol,petriny,green).
connected(petriny,nadrazi_veleslavin,green).
connected(nadrazi_veleslavin,borislavka,green).
connected(borislavka,dejvicka,green).
connected(dejvicka,hradcanska,green).
connected(hradcanska,malostranska,green).
connected(malostranska,staromestska,green).
connected(staromestska,mustek,green).
connected(mustek,muzeum,green).
connected(muzeum,namesti_miru,green).
connected(namesti_miru,jiriho_z_podebrad,green).
connected(jiriho_z_podebrad,flora,green).
connected(flora,zelivskeho,green).
connected(zelivskeho,strasnicka,green).
connected(strasnicka,skalka,green).
connected(skalka,depo_hostivar,green).
```

A Prolog DB (2)

```
connected(letnany,prosek,red) .  
connected(prosek,strizkov,red) .  
connected(strizkov,ladvi,red) .  
connected(ladvi,kobylisy,red) .  
connected(kobylisy,nadrazi_holesovice,red) .  
connected(nadrazi_holesovice,vltavska,red) .  
connected(vltavska,florenc,red) .  
connected(florenc,hlavni_nadrazi,red) .  
connected(hlavni_nadrazi,muzeum,red) .  
connected(muzeum,i_p_pavlova,red) .  
connected(i_p_pavlova,vysehrad,red) .  
connected(vysehrad,prazskeho_povstani,red) .  
connected(prazskeho_povstani,pankrac,red) .  
connected(pankrac,budejovicka,red) .  
connected(budejovicka,kacerov,red) .  
connected(kacerov,roztyly,red) .  
connected(roztyly,chodov,red) .  
connected(chodov,opatov,red) .  
connected(opatov,haje,red) .
```

A Prolog DB (3)

```
connected(zlicin,stodulky,yellow).
connected(stodulky,luka,yellow).
connected(luka,luziny,yellow).
connected(luziny,hurka,yellow).
connected(hurka,nove_butovice,yellow).
connected(nove_butovice,jinonice,yellow).
connected(jinonice,radlicka,yellow).
connected(radlicka,smichov,yellow).
connected(smichov,andel,yellow).
connected(andel,karlovo_namesti,yellow).
connected(karlovo_namesti,narodni_trida,yellow).
connected(narodni_trida,mustek,yellow).
connected(mustek,namesti_republiky,yellow).
connected(namesti_republiky,florenc,yellow).
connected(florenc,krizikova,yellow).
connected(krizikova,invalidovna,yellow).
connected(invalidovna,palmovka,yellow).
connected(palmovka,ceskomoravska,yellow).
connected(ceskomoravska,vysocanska,yellow).
connected(vysocanska,kolbenova,yellow).
connected(kolbenova,hlobetin,yellow).
connected(hlobetin,rajska_zahrada,yellow).
connected(rajska_zahrada,cerny_most,yellow).
```

“Nearby”

Two stations are nearby if they are on the same line with at most one other station in between:

“Nearby”

Two stations are nearby if they are on the same line with at most one other station in between:

```
nearby(zlicin,luka) .  
nearby(luka,zlicin) .  
nearby(zlicin,stodulky) .  
nearby(stodulky,zlicin) .  
nearby(luka,luziny) .  
nearby(luziny,luka) .  
nearby(luka,hurka) .
```

“Nearby”

Two stations are nearby if they are on the same line with at most one other station in between:

```
nearby(zlicin,luka) .  
nearby(luka,zlicin) .  
nearby(zlicin,stodulky) .  
nearby(stodulky,zlicin) .  
nearby(luka,luziny) .  
nearby(luziny,luka) .  
nearby(luka,hurka) .  
...
```

or better

```
nearby(X,Y):-connectedS(X,Y,L) .  
nearby(X,Y):-connectedS(X,Z,L),connectedS(Z,Y,L) .  
connectedS(X,Y,W) :- connected(X,Y,W) .  
connectedS(X,Y,W) :- connected(Y,X,W) .
```

“Not too far”

Compare

```
nearby (X, Y) :- connectedS (X, Y, L) .
```

```
nearby (X, Y) :- connectedS (X, Z, L) , connectedS (Z, Y, L) .
```

with

```
not_too_far (X, Y) :- connectedS (X, Y, L) .
```

```
not_too_far (X, Y) :- connectedS (X, Z, L1) , connectedS (Z, Y, L2) .
```

“Not too far”

Compare

```
nearby (X, Y) :- connectedS (X, Y, L) .
```

```
nearby (X, Y) :- connectedS (X, Z, L) , connectedS (Z, Y, L) .
```

with

```
not_too_far (X, Y) :- connectedS (X, Y, L) .
```

```
not_too_far (X, Y) :- connectedS (X, Z, L1) , connectedS (Z, Y, L2) .
```

This can be rewritten with don't cares:

```
not_too_far (X, Y) :- connectedS (X, Y, _) .
```

```
not_too_far (X, Y) :- connectedS (X, Z, _) , connectedS (Z, Y, _) .
```

?-nearby(mustek, **W)**

?-nearby(mustek,W)

nearby(X1,Y1):-connectedS(X1,Y1,L1)

?-nearby(mustek,W)



nearby(X1,Y1):-connectedS(X1,Y1,L1)

{X1->mustek, Y1->W}

?-connectedS(mustek,W,L1)

?-nearby(mustek,W)



nearby(X1,Y1):-connectedS(X1,Y1,L1)

{X1->mustek, Y1->W}

?-connectedS(mustek,W,L1)

connectedS(X,Y,L) :- connected(X,Y,L)

?-nearby(mustek,W)

nearby(X1,Y1):-connectedS(X1,Y1,L1)
{X1->mustek, Y1->W}

?-connectedS(mustek,W,L1)

connectedS(X,Y,L) :- connected(X,Y,L)
{X->mustek, Y->W, L->L1}

?-connected(mustek,W,L1)

?-nearby(mustek,W)



nearby(X1,Y1):-connectedS(X1,Y1,L1)

{X1->mustek, Y1->W}

?-connectedS(mustek,W,L1)



connectedS(X,Y,L) :- connected(X,Y,L)

{X->mustek, Y->W, L->L1}

?-connected(mustek,W,L1)

connected(mustek,muzeum,green)

?-nearby(mustek,W)

nearby(X1,Y1):-connectedS(X1,Y1,L1)

{X1->mustek, Y1->W}

?-connectedS(mustek,W,L1)

connectedS(X,Y,L) :- connected(X,Y,L)

{X->mustek, Y->W, L->L1}

?-connected(mustek,W,L1)

connected(mustek,muzeum,green)

{W->muzeum, L1->green}

[]

“Reachable”

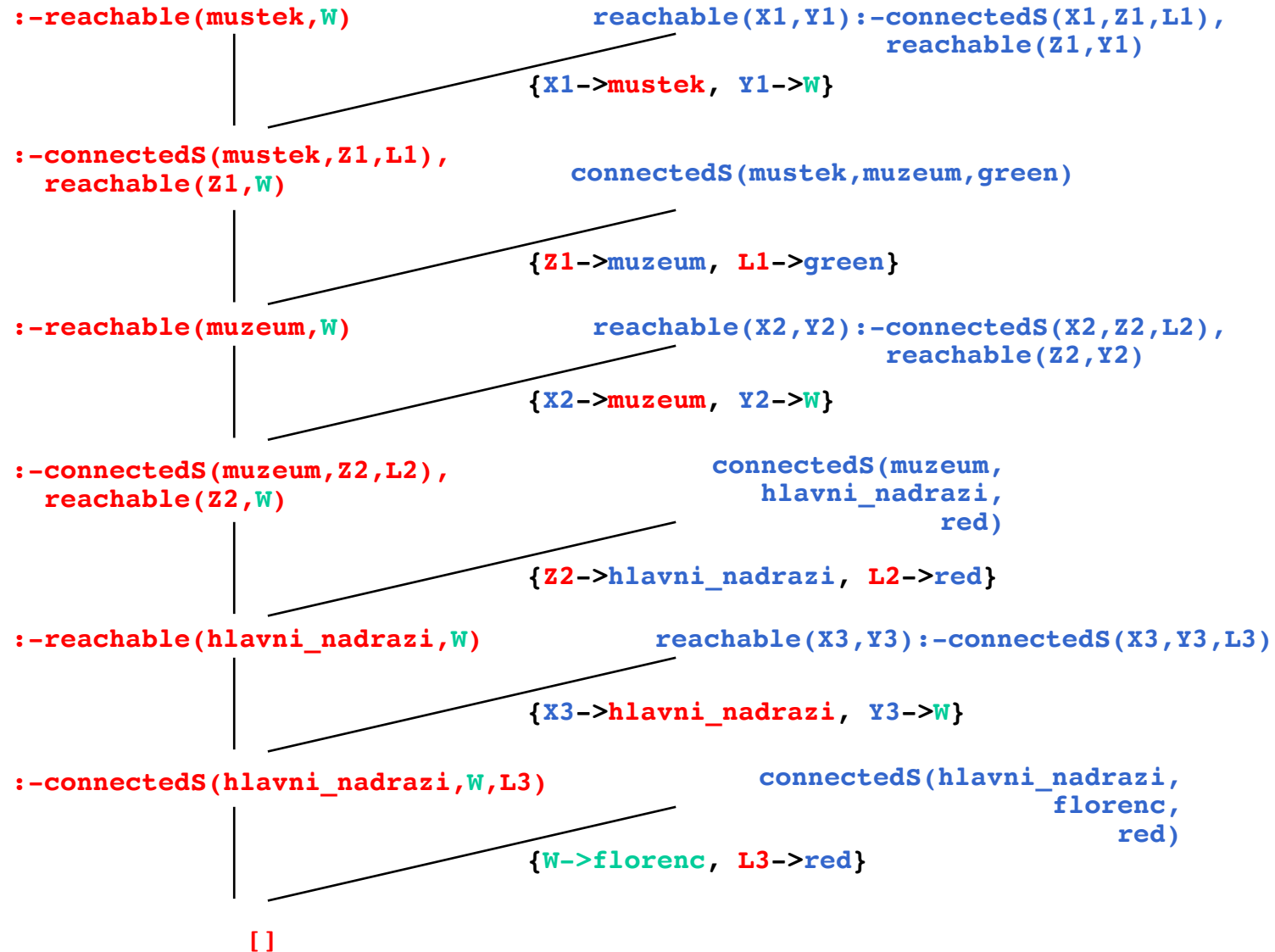
A station is reachable from another if they are on the same line, or with one, two, ... changes:

```
reachable(X,Y):-connectedS(X,Y,L) .  
reachable(X,Y):-connectedS(X,Z,L1),connectedS(Z,Y,L2) .  
reachable(X,Y):-connectedS(X,Z1,L1),connectedS(Z1,Z2,L2),  
                    connectedS(Z2,Y,L3) .  
...
```

or better

```
reachable(X,Y):-connectedS(X,Y,L) .  
reachable(X,Y):-connectedS(X,Z,L),reachable(Z,Y) .
```

Note: To make the example more compact, we assume that we have the predicate `connectedS` given as ground facts.



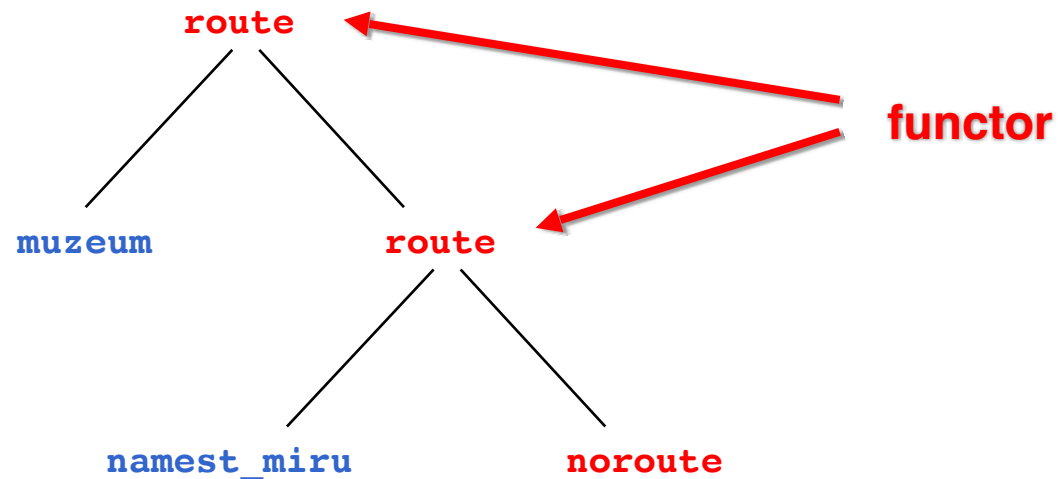
There is a catch!

- The answers that we get depend on the exact way Prolog works inside. We will talk about that next time.

“Recording the Path”

```
reachable(X,Y,noroute):-connected(X,Y,L).  
reachable(X,Y,route(Z,R)):-connected(X,Z,L),  
                                reachable(Z,Y,R).
```

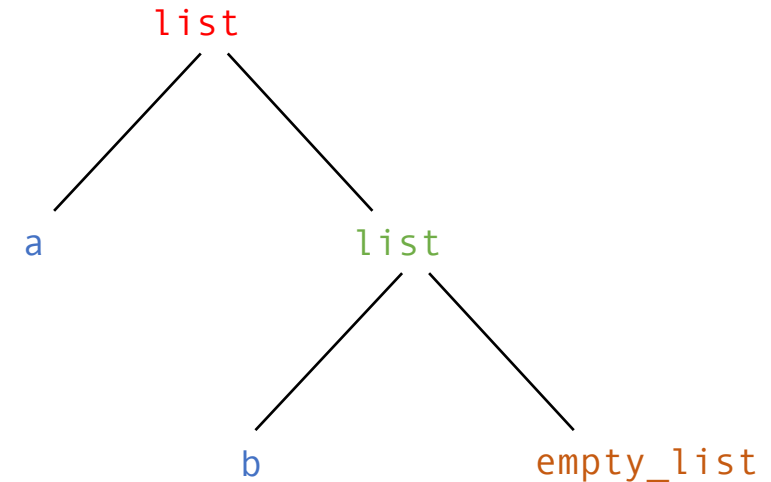
```
?-reachable(mustek,jiriho_z_podebrad,R).  
R = route(muzeum,route(namesti_miru,noroute));  
...
```



More examples (Still with our “own” lists)

- Here we will represent lists more or less like we represented routes (Prolog has built in support for lists but this is still to illustrate them as pure logic concepts), e.g.: `empty_list`, `list(X,empty_list)`,

`list(a,list(b,empty_list))`



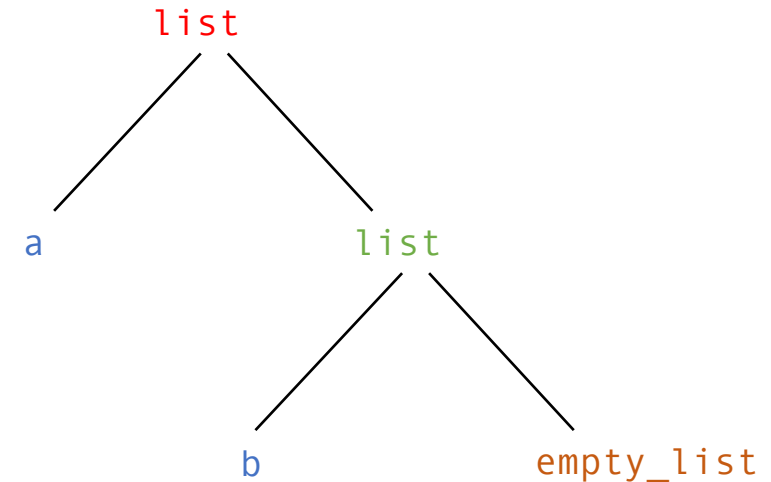
More examples (Still with our “own” lists)

- Here we will represent lists more or less like we represented routes (Prolog has built in support for lists but this is still to illustrate them as pure logic concepts), e.g.: `empty_list`, `list(X,empty_list)`,

Example:

```
add_to_start(empty_list, X, list(X,empty_list)).  
add_to_start(list(H,T), X, list(X,list(H,T))).
```

`list(a,list(b,empty_list))`



swish.swi-prolog.org

SWISH -- SWI-Prolog for SHaring

 **SWISH**

File Edit Examples Help

 251 users online

Search

Program

```
1 add_to_start(empty_list, X, list(X,empty_list)).
2 add_to_start(list(H,T), X, list(X,list(H,T))).
3
4
```

 `add_to_start(empty_list, a, L1), add_to_start(L1, b, L2),`
`add_to_start(L2, c, L3)`

`L1 = list(a, empty_list),`
`L2 = list(b, list(a, empty_list)),`
`L3 = list(c, list(b, list(a, empty_list)))`

?-

`add_to_start(empty_list, a, L1), add_to_start(L1, b,`
`L2), add_to_start(L2, c, L3)`

Examples History Solutions

☐ table results **Run!**

swish.swi-prolog.org

SWISH -- SWI-Prolog for SHaring

**SWISH**

File Edit Examples Help

251 users online

Search



Program

```
1 add_to_start(empty_list, X, list(X,empty_list)).
2 add_to_start(list(H,T), X, list(X,list(H,T))).
3
4
```



 add_to_start(empty_list, a, L1), add_to_start(L1, b, L2),
add_to_start(L2, c, L3)

L1 = list(a, empty_list),
L2 = list(b, list(a, empty_list)),
L3 = list(c, list(b, list(a, empty_list)))

?-

add_to_start(empty_list, a, L1), add_to_start(L1, b, L2), add_to_start(L2, c, L3)

Examples History Solutions

☐ table results **Run!**

swish.swi-prolog.org

SWISH -- SWI-Prolog for SHaring

**SWISH**

File Edit Examples Help

251 users online

Search



Program

```
1 add_to_start(empty_list, X, list(X,empty_list)).
2 add_to_start(list(H,T), X, list(X,list(H,T))).
3
4
```

 add_to_start(empty_list, a, L1), add_to_start(L1, b, L2),
add_to_start(L2, c, L3)

L1 = list(a, empty_list),
L2 = list(b, list(a, empty_list)),
L3 = list(c, list(b, list(a, empty_list)))

?-

add_to_start(empty_list, a, L1), add_to_start(L1, b,
L2), add_to_start(L2, c, L3)

Examples History Solutions

☐ table results **Run!**

Yet another example

```
add_to_end(empty_list, X, list(X,empty_list)).
```

```
add_to_end(list(H,T), X, list(H,T2)) :- add_to_end(T,X,T2).
```

swish.swi-prolog.org

SWISH -- SWI-Prolog for SHaring

**SWISH**

File

Edit

Examples

Help



244 users online

Search

Q





25

Program

1

2

3

4

5

```
add_to_end(empty_list, X, list(X,empty_list)).
add_to_end(list(H,T), X, list(H,T2)) :- add_to_end(T,
```



```
add_to_end(empty_list, a, L1), add_to_end(L1, b, L2),
add_to_end(L2, c, L3)
```

```
L1 = list(a, empty_list),
L2 = list(a, list(b, empty_list)),
L3 = list(a, list(b, list(c, empty_list)))
```

?-

```
add_to_end(empty_list, a, L1), add_to_end(L1, b, L2),
add_to_end(L2, c, L3)
```

Examples

History

Solutions

☐ table results

Run!

Open #tabbed-tab-0 on this page in a new tab

swish.swi-prolog.org

SWISH -- SWI-Prolog for SHaring

 **SWISH**

File Edit Examples Help

 244 users online

Search



Program

```
1 add_to_end(empty_list, X, list(X,empty_list)).
2 add_to_end(list(H,T), X, list(H,T2)) :- add_to_end(T,
3
4
5
```

 add_to_end(empty_list, a, L1), add_to_end(L1, b, L2),
add_to_end(L2, c, L3)

L1 = list(a, empty_list),
L2 = list(a, list(b, empty_list)),
L3 = list(a, list(b, list(c, empty_list)))

?-

add_to_end(empty_list, a, L1), add_to_end(L1, b, L2),
add_to_end(L2, c, L3)

Examples History Solutions

☐ table results

Run!

Open #tabbed-tab-0 on this page in a new tab

swish.swi-prolog.org

SWISH -- SWI-Prolog for SHaring

 **SWISH**

File Edit Examples Help

 244 users online



Program

```
1 add_to_end(empty_list, X, list(X,empty_list)).
2 add_to_end(list(H,T), X, list(H,T2)) :- add_to_end(T,
3
4
5
```

 add_to_end(empty_list, a, L1), add_to_end(L1, b, L2),
add_to_end(L2, c, L3)

L1 = list(a, empty_list),
L2 = list(a, list(b, empty_list)),
L3 = list(a, list(b, list(c, empty_list)))

?-

```
add_to_end(empty_list, a, L1), add_to_end(L1, b, L2),  
add_to_end(L2, c, L3)
```

Examples History Solutions

☐ table results

Open #tabbed-tab-0 on this page in a new tab

Removing last element

We had:

```
add_to_end(empty_list, X, list(X,empty_list)).  
add_to_end(list(H,T), X, list(H,T2)) :- add_to_end(T,X,T2).
```

We can use:

```
remove_last(List,NewList) :- add_to_end(NewList,_,List).
```

Removing last element

We had:

```
add_to_end(empty_list, X, list(X,empty_list)).  
add_to_end(list(H,T), X, list(H,T2)) :- add_to_end(T,X,T2).
```

in *out*

We can use:

```
remove_last(List,NewList) :- add_to_end(NewList,_,List).
```

out *in*

swish.swi-prolog.org

SWISH -- SWI-Prolog for SHaring

SWISH

File

Edit

Examples

Help

251 users online

Search

25

Program

1

2

3

4

5

6

7

add_to_end(empty_list, X, list(X,empty_list)).

add_to_end(list(H,T), X, list(H,T2)) :- add_to_end(T,X,T2).

remove_last(List,NewList) :- add_to_end(NewList,_,List).

add_to_end(empty_list, a, L1),

add_to_end(L1, b, L2), add_to_end(L2, c, L3),

remove_last(L3,L4)

L1 = list(a, empty_list),

L2 = L4, L4 = list(a, list(b, empty_list)),

L3 = list(a, list(b, list(c, empty_list)))

Next

10

100

1,000

Stop

?-

add_to_end(empty_list, a, L1),

add_to_end(L1, b, L2), add_to_end(L2, c,

L3), remove_last(L3,L4)|

Examples

History

Solutions

☐ table results

Run!

swish.swi-prolog.org

SWISH -- SWI-Prolog for SHaring

**SWISH**

File Edit Examples Help

 251 users online



Program

```
1 add_to_end(empty_list, X, list(X,empty_list)).
2 add_to_end(list(H,T), X, list(H,T2)) :- add_to_end(T,X,T2).
3
4 remove_last(List,NewList) :- add_to_end(NewList,_,List).
5
6
7
```

 add_to_end(empty_list, a, L1),
add_to_end(L1, b, L2), add_to_end(L2, c, L3),
remove_last(L3,L4)

L1 = list(a, empty_list),
L2 = L4, L4 = list(a, list(b, empty_list)),
L3 = list(a, list(b, list(c, empty_list)))

Next101001,000Stop

?-

add_to_end(empty_list, a, L1),
add_to_end(L1, b, L2), add_to_end(L2, c,
L3), remove_last(L3,L4)|

ExamplesHistorySolutions

☐ table results

Run!

swish.swi-prolog.org

SWISH -- SWI-Prolog for SHaring

**SWISH**

File Edit Examples Help

251 users online

Search



Program

```
1 add_to_end(empty_list, X, list(X,empty_list)).
2 add_to_end(list(H,T), X, list(H,T2)) :- add_to_end(T,X,T2).
3
4 remove_last(List,NewList) :- add_to_end(NewList,_,List).
5
6
7
```

add_to_end(empty_list, a, L1),
add_to_end(L1, b, L2), add_to_end(L2, c, L3),
remove_last(L3,L4)

L1 = list(a, empty_list),
L2 = L4, L4 = list(a, list(b, empty_list)),
L3 = list(a, list(b, list(c, empty_list)))

Next101001,000Stop

?-

add_to_end(empty_list, a, L1),
add_to_end(L1, b, L2), add_to_end(L2, c,
L3), remove_last(L3,L4)|

ExamplesHistorySolutions

☐ table results

Run!

Built-in Lists in Prolog

Examples of lists:

`[]`, `[a,b,c]` , `[a,X,c,d]` , ...

Instead of writing `list(H,T)`, we write `[H|T]` in Prolog.

```
add_to_end(empty_list, X, list(X,empty_list)).
```

```
add_to_end(list(H,T), X, list(H,T2)) :- add_to_end(T,X,T2).
```

```
add_to_end([], X, [X]).
```

```
add_to_end([H|T], X, [H|T2]) :- add_to_end(T,X,T2).
```

swish.swi-prolog.org

SWISH -- SWI-Prolog for SHaring

SWISH

File

Edit

Examples

Help

240 users online

Search

25

Program

1

2

3

4

5

6

7

8

add_to_end([], X, [X]).

add_to_end([H|T], X, [H|T2]) :- add_to_end(T,X,T2).

remove_last(List,NewList) :- add_to_end(NewList,_,List).

add_to_end([], a, L1),

add_to_end(L1, b, L2),

remove_last(L3,L4)

L1 = list(a, empty_list),

L2 = L4, L4 = list(a, list(b, empty_list)),

L3 = list(a, list(b, list(c, empty_list)))

add_to_end([], a, L1), add_to_end(L1, b, L2),

add_to_end(L2, c, L3), remove_last(L3,L4)

L1 = [a],

L2 = L4, L4 = [a, b],

L3 = [a, b, c]

Next101001,000Stop

?-

add_to_end([], a, L1), add_to_end(L1, b,

L2), add_to_end(L2, c, L3),

remove_last(L3,L4)

ExamplesHistorySolutions

☐ table results

Run!

swish.swi-prolog.org

SWISH -- SWI-Prolog for SHaring

SWISH

File

Edit

Examples

Help

240 users online

Search

25

Program

1

2

3

4

5

6

7

8

add_to_end([], X, [X]).

add_to_end([H|T], X, [H|T2]) :- add_to_end(T,X,T2).

remove_last(List,NewList) :- add_to_end(NewList,_,List).

add_to_end(empty_list, a, L1),

add_to_end(L1, b, L2), add_to_end(L2, c, L3),

remove_last(L3,L4)

L1 = list(a, empty_list),

L2 = L4, L4 = list(a, list(b, empty_list)),

L3 = list(a, list(b, list(c, empty_list)))

add_to_end([], a, L1), add_to_end(L1, b, L2),

add_to_end(L2, c, L3), remove_last(L3,L4)

L1 = [a],

L2 = L4, L4 = [a, b],

L3 = [a, b, c]

Next

10

100

1,000

Stop

?-

add_to_end([], a, L1), add_to_end(L1, b,

L2), add_to_end(L2, c, L3),

remove_last(L3,L4)

Examples

History

Solutions

☐ table results

Run!

SLD Resolution in Detail

Unification

Recap from the last lecture... Substitutions

- **Example:**

`likes(X,Y) :- dogPerson(X), dog(Y).`

Recap from the last lecture... Substitutions

- **Example:**

`likes(X,Y) :- dogPerson(X), dog(Y).`

`S = {X/alice, Y/barney}`

Recap from the last lecture... Substitutions

- **Example:**

`likes(X,Y) :- dogPerson(X), dog(Y).`

`S = {X/alice, Y/barney}`

`likes(alice, barney) :- dogPerson(alice), dog(barney)`

Unification

- (We need unification to perform resolution.)
- **Unifier:** Given two atoms or terms A and B with disjoint (!) sets of variables, their **unifier** is a substitution θ such that $A\theta = B\theta$.

Unification

- (We need unification to perform resolution.)
- **Unifier:** Given two atoms or terms A and B with disjoint (!) sets of variables, their **unifier** is a substitution Θ such that $A\Theta = B\Theta$.
- A unifier Θ is more general (*technically speaking, we should be saying more or equally general*) than a unifier Θ' iff there exists a substitution σ such that $\Theta' = \Theta \sigma$.

Unification - Example

- ...A unifier Θ is more general (*technically speaking, we should be saying more or equally general*) than a unifier Θ' iff there exists a substitution σ such that $\Theta' = \Theta \sigma$. If Θ is more general than Θ' and not vice versa, then Θ is strictly more general than Θ' .

Example:

$A = \text{studies}(X, Y)$

$B = \text{studies}(\text{alice}, Z)$

$\Theta = \{X/\text{alice}, Y/Z\}$

$\Theta' = \{X/\text{alice}, Y/\text{bob}, Z/\text{bob}\}$

$\Theta' = \Theta \sigma$, where $\sigma = \{Z/\text{bob}\}$

Unification – Most General Unifier

- **Most General Unifier:** Given two atoms or terms A and B with disjoint sets of variables, their **most general unifier (MGU)** is a unifier θ such that there is no strictly more general unifier of A and B .

- **Example:**

$A = \text{studies}(X, Y)$

$B = \text{studies}(\text{alice}, Z)$

$\theta = \{X/\text{alice}, Y/Z\}$

Here, θ is an MGU of A and B .

The Hebrand Unification Algorithm

(Adapted from Pierre M. Nugues: Language Processing with Perl and Prolog)

- **Initialization:** $\Theta = \{\}$, `failure = false`, **push** $A = B$ on the stack

- **Loop:**

- 1: **Pop** $x = y$ from the stack

- 2: **If** x and y are constants and $x == y$ **then continue.**

- 3: **Else if** x is a variable and x does not appear in y **then** /* “Occurs check” */

- 4: Replace x with y in the stack and in Θ . Add the substitution x/y to Θ .

- 5: **Else if** x is a variable and $x == y$ **then continue.**

- 6: **Else if** y is a variable and x is not a variable **then push** $y = x$ on the stack.

- 7: **Else if** x and y are compound and x is $f(t_1, \dots, t_k)$ and y is $f(u_1, \dots, u_k)$ **then**

- 8: **Push** on the stack: $t_1 = u_1$, $t_2 = u_2$, ..., $t_k = u_k$.

- 9: **Else set** `failure = true`, $\Theta = \{\}$ and **return.**

Until stack is empty.

Example

$$f(g(X, h(X, b)), Z) = f(g(a, Z), Y)$$

(Adapted from Pierre M. Nugues: Language Processing with Perl and Prolog)

Stack:

$f(g(X, h(X, b)), Z) = f(g(a, Z), Y)$ */* In Prolog, "=" means "unify" */*

$\Theta = \{\}$

Example

$$f(g(X, h(X, b)), Z) = f(g(a, Z), Y)$$

(Adapted from Pierre M. Nugues: Language Processing with Perl and Prolog)

Stack:

$$g(X, h(X, b)) = g(a, Z)$$

$$Z = Y$$

$$\Theta = \{\}$$

Example

$$f(g(X, h(X, b)), Z) = f(g(a, Z), Y)$$

(Adapted from Pierre M. Nugues: Language Processing with Perl and Prolog)

Stack:

$$X = a$$

$$h(X, b) = Z$$

$$Z = Y$$

$$\Theta = \{\}$$

Example

$$f(g(X, h(X, b)), Z) = f(g(a, Z), Y)$$

(Adapted from Pierre M. Nugues: Language Processing with Perl and Prolog)

Stack:

$$h(a, b) = Z$$

$$Z = Y$$

$$\Theta = \{X/a\}$$

Example

$$f(g(X, h(X, b)), Z) = f(g(a, Z), Y)$$

(Adapted from Pierre M. Nugues: Language Processing with Perl and Prolog)

Stack:

$$Z = h(a, b)$$

$$Z = Y$$

$$\Theta = \{X/a\}$$

Example

$$f(g(X, h(X, b)), Z) = f(g(a, Z), Y)$$

(Adapted from Pierre M. Nugues: Language Processing with Perl and Prolog)

Stack:

$$h(a, b) = Y$$

$$\Theta = \{X/a, Z/h(a, b)\}$$

Example

$$f(g(X, h(X, b)), Z) = f(g(a, Z), Y)$$

(Adapted from Pierre M. Nugues: Language Processing with Perl and Prolog)

Stack:

EMPTY

$$\Theta = \{X/a, Z/h(a, b), Y/h(a, b)\}$$

Result of unification - MGU:

$$f(g(a, h(a, b)), h(a, b))$$

Example

$$f(g(X, h(X, b)), Z) = f(g(a, Z), Y)$$

The screenshot shows the SWISH web interface at swish.swi-prolog.org. The interface includes a top navigation bar with "File", "Edit", "Examples", and "Help" menus, a search bar, and a notification bell showing 25 alerts. Below the navigation bar, there is a "Program" tab with a plus icon. The main area displays a Prolog query and its solution, which is highlighted with a red circle. The query is $f(g(X, h(X, b)), Z) = f(g(a, Z), Y)$. The solution shows $X = a$ and $Y = Z, Z = h(a, b)$. Below the solution, there is a query prompt $?- f(g(X, h(X, b)), Z) = f(g(a, Z), Y)$. At the bottom, there are buttons for "Examples", "History", "Solutions", a checkbox for "table results", and a "Run!" button.

SWISH -- SWI-Prolog for Sharing

199 users online

Program

1
2
3
4

$f(g(X, h(X, b)), Z) = f(g(a, Z), Y)$

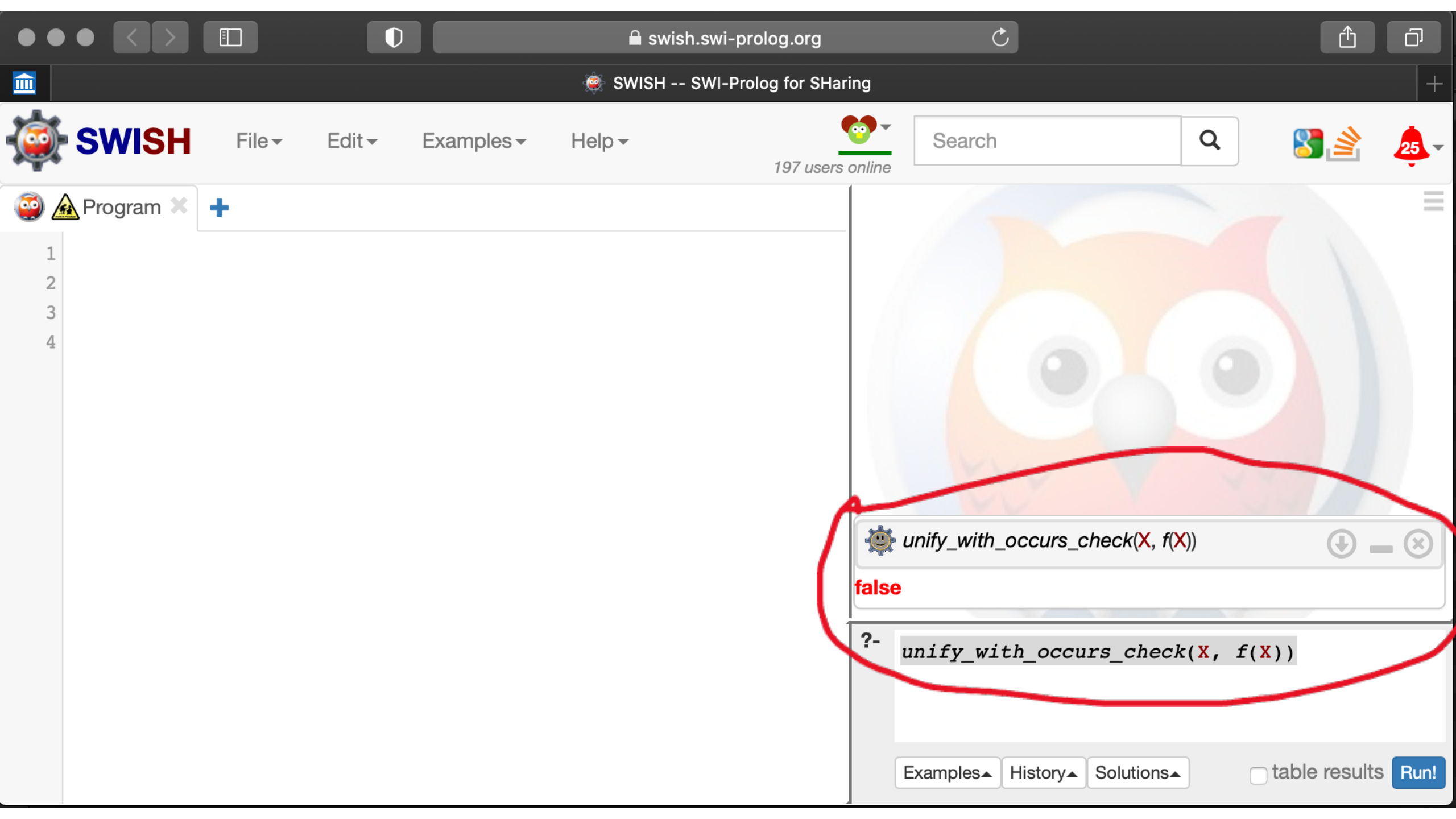
$X = a,$
 $Y = Z, Z = h(a, b)$

?- $f(g(X, h(X, b)), Z) = f(g(a, Z), Y)$

Examples History Solutions ☐ table results Run!

The “Occurs Check”

- Line 3: **Else if** x is a variable and x does not appear in y **then** ...
- This is not implemented in many Prologs for efficiency reasons and programmers have to take care of it themselves. Without occurs check, Prolog inference is not really sound (this is a tradeoff for efficiency).
- If you want sound unification, use `unify_with_occurs_check/2`



swish.swi-prolog.org

SWISH -- SWI-Prolog for SHaring

**SWISH**

File Edit Examples Help

 195 users online

Search



Program

1
2
3
4

 unify_with_occurs_check(**X**, f(**X**))

false

 **X** = f(**X**)

X = f(**X**)

?- **X** = f(**X**)

Examples History Solutions

☐ table results

Run!

Next week... SLD trees