Transforming Coroutining Logic Programs into Equivalent CHR Programs

Vincent Nys
Danny De Schreye

KU Leuven

29 April 2017

Concept

- Variant of Compiling Control (CC)

Concept

- Variant of Compiling Control (CC)
- Original: LP $+$ non-standard rule $\rightarrow$ Prolog

Concept

- Variant of Compiling Control (CC)
- Original: LP + non-standard rule → Prolog
- Variant: LP + non-standard rule → CHR

Motivation

- Analysis tools for coroutining LP are rare

Motivation

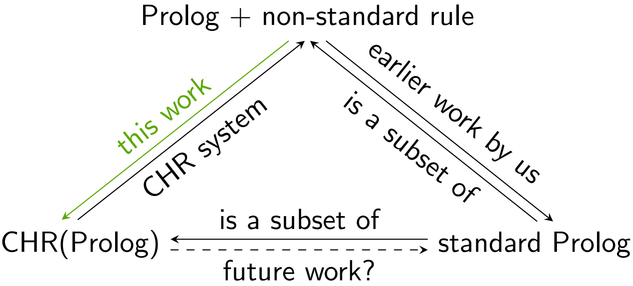
- Analysis tools for coroutining LP are rare
- More natural execution model and syntax

Motivation

- Analysis tools for coroutining LP are rare
- More natural execution model and syntax
- Portability (CHR in C, Java, JS, Haskell,. . .)

Motivation

- Analysis tools for coroutining LP are rare
- More natural execution model and syntax
- Portability (CHR in C, Java, JS, Haskell,. . .)



CHR example [Schrijvers, 2008]

```
red,blue <=> purple.
red,yellow <=> orange.
blue,yellow <=> green.
```

CHR example [Schrijvers, 2008]

```
red,blue <=> purple.
red,yellow <=> orange.
blue,yellow <=> green.
```

```
?- red, yellow, blue.
```

$\mathcal{S} = \emptyset$

CHR example [Schrijvers, 2008]

```
red,blue <=> purple.
red,yellow <=> orange.
blue,yellow <=> green.
```

```
?- yellow, blue.
```

$\mathcal{S} = \{\texttt{red}\}$

CHR example [Schrijvers, 2008]

```
red,blue <=> purple.
red,yellow <=> orange.
blue,yellow <=> green.
```

```
?- blue.
```

$\mathcal{S} = \{\texttt{red, yellow}\}$

CHR example [Schrijvers, 2008]

```
red,blue <=> purple.
red,yellow <=> orange.
blue,yellow <=> green.
```

```
?- blue.
```

$\mathcal{S} = \{\texttt{orange}\}$

CHR example [Schrijvers, 2008]

```
red,blue <=> purple.
red,yellow <=> orange.
blue,yellow <=> green.
```

```
?-.
```

$\mathcal{S} = \{\texttt{orange, blue}\}$

Permutation sort

```
sort(X,Y) :- perm(X,Y), ord(Y).
perm([],[]).
perm([X|Y],[U|V]) :-
  del(U,[X|Y],W),
  perm(W,V).
ord([]).
ord([X]).
ord([X,Y|Z]) :- X =< Y, ord([Y|Z]).
```

ACPD: absolute essentials

- $a_i, i \in \mathbb{N}_0$

ACPD: absolute essentials

- $a_i, i \in \mathbb{N}_0$
- $g_j, j \in \mathbb{N}_0$

ACPD: absolute essentials

- $a_i, i \in \mathbb{N}_0$
- $g_j, j \in \mathbb{N}_0$
- abstract atoms, functions, conjunctions

ACPD: absolute essentials

- $a_i, i \in \mathbb{N}_0$
- $g_j, j \in \mathbb{N}_0$
- abstract atoms, functions, conjunctions
- concretization function γ

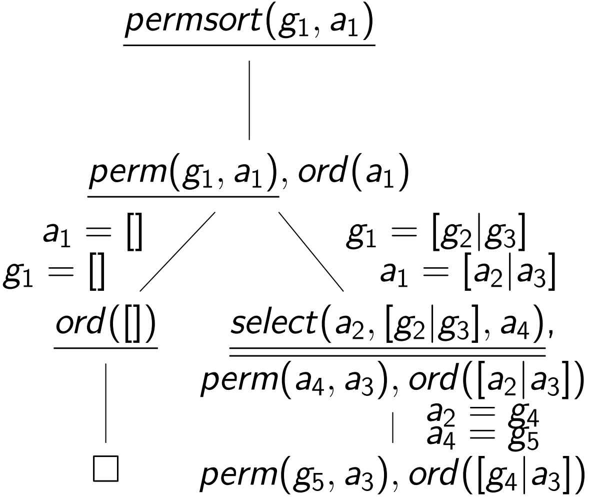
ACPD: absolute essentials

- $a_i, i \in \mathbb{N}_0$
- $g_j, j \in \mathbb{N}_0$
- abstract atoms, functions, conjunctions
- concretization function γ
- abstract resolution

ACPD: absolute essentials

- $a_i, i \in \mathbb{N}_0$
- $g_j, j \in \mathbb{N}_0$
- abstract atoms, functions, conjunctions
- concretization function γ
- abstract resolution
- fixpoint for $\mathcal{A}$

First analysis tree



$\underline{permsort(g_1, a_1)}$

$\underline{perm(g_1, a_1)}, ord(a_1)$

$a_1 = []$
$g_1 = []$

$g_1 = [g_2|g_3]$
$a_1 = [a_2|a_3]$

$\underline{ord([])}$

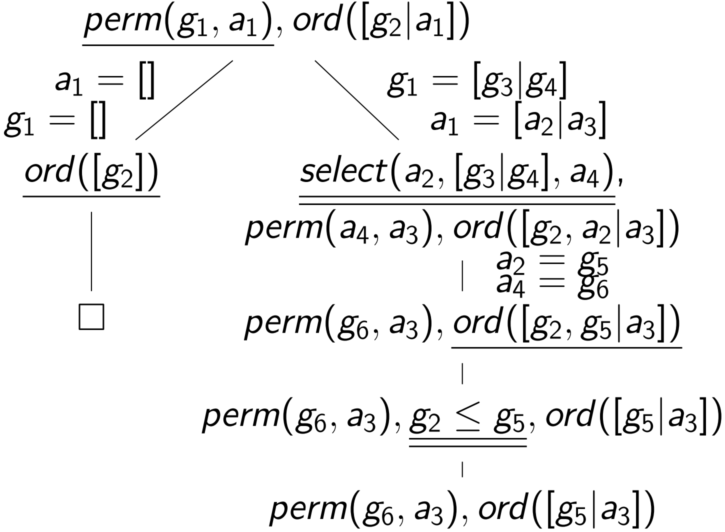
$\square$

$\underline{\underline{select(a_2, [g_2|g_3], a_4)}},$
$perm(a_4, a_3), ord([a_2|a_3])$

$a_2 = g_4$
$a_4 = g_5$

$perm(g_5, a_3), ord([g_4|a_3])$

Second analysis tree



$\underline{perm(g_1, a_1)}, ord([g_2|a_1])$

$a_1 = []$

$g_1 = []$

$\underline{ord([g_2])}$

$\square$

$g_1 = [g_3|g_4]$

$a_1 = [a_2|a_3]$

$\underline{\underline{select(a_2, [g_3|g_4], a_4)}},$

$perm(a_4, a_3), ord([g_2, a_2|a_3])$

$a_2 = g_5$

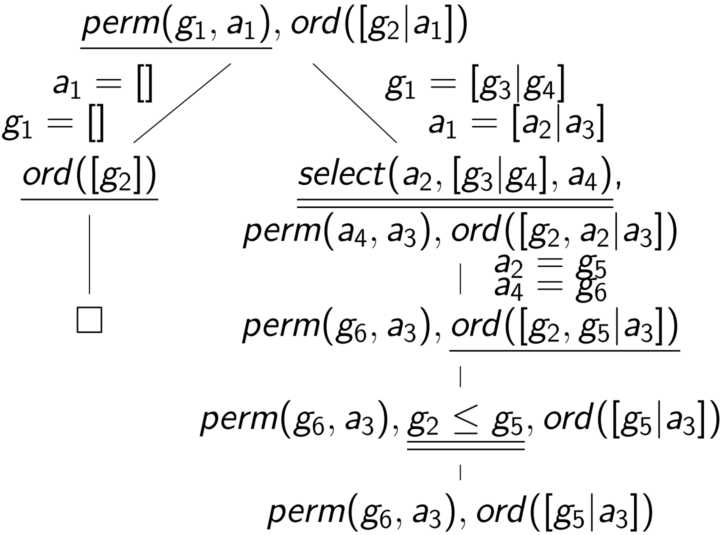
$a_4 = g_6$

$perm(g_6, a_3), \underline{ord([g_2, g_5|a_3])}$

$perm(g_6, a_3), \underline{\underline{g_2 \leq g_5}}, ord([g_5|a_3])$

$perm(g_6, a_3), ord([g_5|a_3])$

Second analysis tree



$\underline{perm(g_1, a_1)}, ord([g_2|a_1])$

$a_1 = []$

$g_1 = []$

$\underline{ord([g_2])}$

$\square$

$g_1 = [g_3|g_4]$

$a_1 = [a_2|a_3]$

$\underline{\underline{select(a_2, [g_3|g_4], a_4)}},$

$perm(a_4, a_3), ord([g_2, a_2|a_3])$

$a_2 = g_5$

$a_4 = g_6$

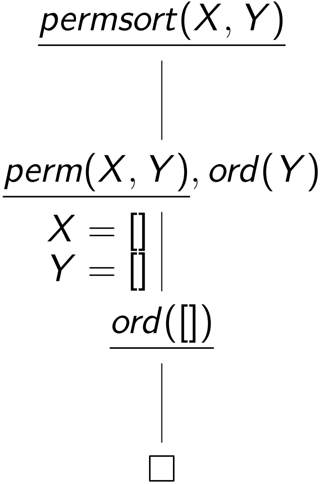
$perm(g_6, a_3), \underline{ord([g_2, g_5|a_3])}$

$perm(g_6, a_3), \underline{\underline{g_2 \leq g_5}}, ord([g_5|a_3])$

$perm(g_6, a_3), ord([g_5|a_3])$

fixpoint for $\mathcal{A}$

First synthesis tree, left branch



$\underline{permsort(X, Y)}$

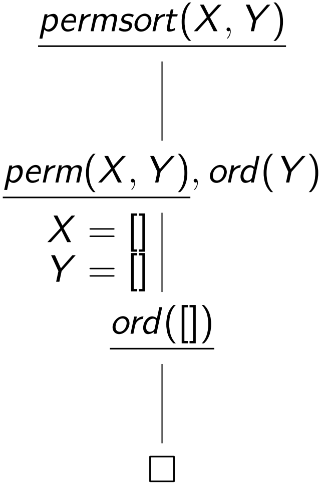
$\underline{perm(X, Y)}, ord(Y)$

$X = []$
$Y = []$

$\underline{ord([])}$

$\square$

First synthesis tree, left branch



$\underline{permsort(X, Y)}$

$\underline{perm(X, Y)}, ord(Y)$

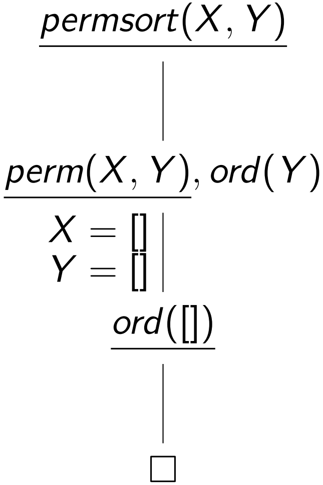
$X = []$
$Y = []$

$\underline{ord([])}$

$\square$

$permsort([], []) \leftarrow true$

First synthesis tree, left branch



$\underline{\mathit{permsort}(X, Y)}$

$\underline{\mathit{perm}(X, Y)}, \mathit{ord}(Y)$

$X = []$
$Y = []$

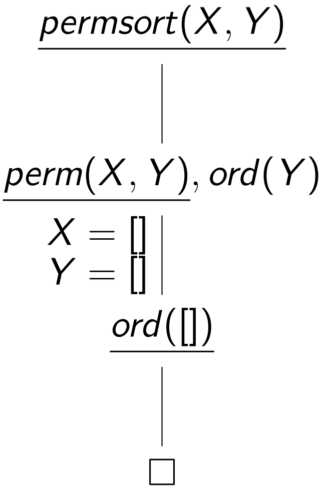
$\underline{\mathit{ord}([])}$

$\square$

$\mathit{permsort}([], []) \leftarrow \mathit{true}$

```
permsort([],[]).
```

First synthesis tree, left branch



$\underline{\mathit{permsort}(X, Y)}$

$\underline{\mathit{perm}(X, Y)}, \mathit{ord}(Y)$

$X = []$
$Y = []$

$\underline{\mathit{ord}([])}$

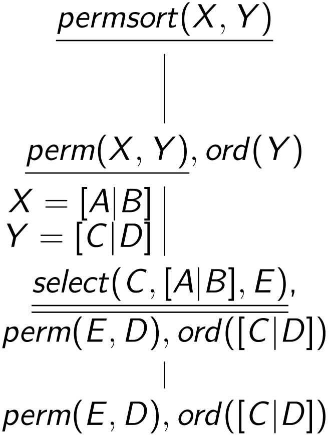
$\square$

$\mathit{permsort}([], []) \leftarrow \mathit{true}$

```
permsort([],[]).
```

```
permsort([],Y) <=> Y = [].
```

First synthesis tree, right branch



$\underline{permsort(X, Y)}$

$\underline{perm(X, Y)}, ord(Y)$

$X = [A|B]$

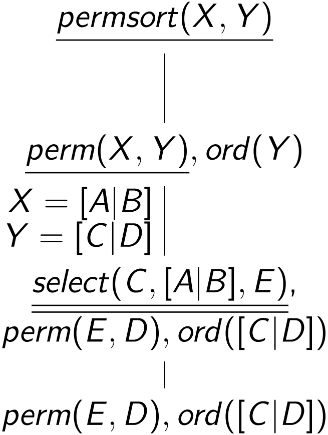
$Y = [C|D]$

$\underline{\underline{select(C, [A|B], E),}}$

$perm(E, D), ord([C|D])$

$perm(E, D), ord([C|D])$

First synthesis tree, right branch



$\underline{permsort(X, Y)}$

$\underline{perm(X, Y)}, ord(Y)$

$X = [A|B]$

$Y = [C|D]$

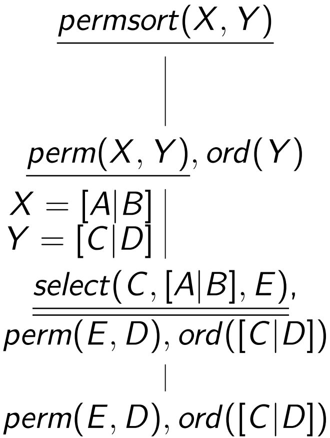
$\underline{select(C, [A|B], E),}$

$\overline{perm(E, D), ord([C|D])}$

$perm(E, D), ord([C|D])$

$$permsort([A|B], [C|D]) \leftarrow$$
$$select(C, [A|B], E) \wedge$$
$$perm(E, D) \wedge ord([C|D])$$

First synthesis tree, right branch



$permsort(X, Y)$

$perm(X, Y), ord(Y)$

$X = [A|B]$
$Y = [C|D]$

$select(C, [A|B], E),$

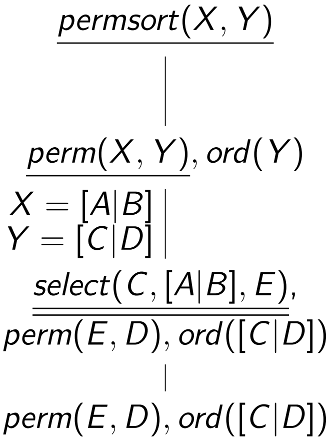
$perm(E, D), ord([C|D])$

$perm(E, D), ord([C|D])$

$$permsort([A|B], [C|D]) \leftarrow select(C, [A|B], E) \wedge perm(E, D) \wedge ord([C|D])$$

```
permsort([A|B],[C|D]) :-
select(C,[A|B],E),
p1(perm(E,D),ord([C|D])).
```

First synthesis tree, right branch



$\underline{\mathit{permsort}(X, Y)}$

$\underline{\mathit{perm}(X, Y)}, \mathit{ord}(Y)$

$X = [A|B]$

$Y = [C|D]$

$\mathit{select}(C, [A|B], E),$

$\mathit{perm}(E, D), \mathit{ord}([C|D])$

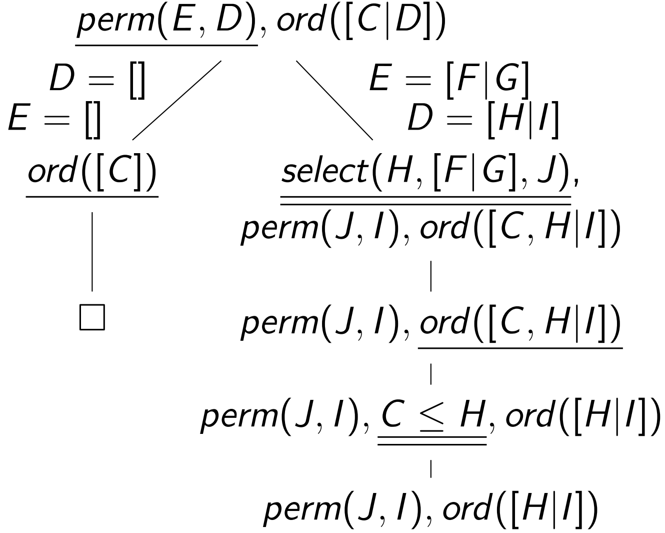
$\mathit{perm}(E, D), \mathit{ord}([C|D])$

$\mathit{permsort}([A|B], [C|D]) \leftarrow$
$\mathit{select}(C, [A|B], E) \wedge$
$\mathit{perm}(E, D) \wedge \mathit{ord}([C|D])$

```
permsort([A|B],[C|D]) :-
select(C,[A|B],E),
p1(perm(E,D),ord([C|D])).
```

```
permsort([A|B],Y) <=>
Y = [C|D],
select(C,[A|B],E),
perm(E,D), ord([C|D]).
```

Second synthesis tree



$\underline{\mathit{perm}(E,D)}, \mathit{ord}([C|D])$

$D = []$

$E = []$

$\underline{\mathit{ord}([C])}$

$\square$

$E = [F|G]$

$D = [H|I]$

$\underline{\underline{\mathit{select}(H,[F|G],J)}},$

$\mathit{perm}(J,I), \mathit{ord}([C,H|I])$

$\mathit{perm}(J,I), \underline{\mathit{ord}([C,H|I])}$

$\mathit{perm}(J,I), \underline{\underline{C \leq H}}, \mathit{ord}([H|I])$

$\mathit{perm}(J,I), \mathit{ord}([H|I])$

Confused queens

```
cqueens(N,D) :-
  genlist(N,L),
  draw(N,L,D),
  confused(D).
draw(0,_,[]).
draw(N,L,[E|R]) :-
  N > 0,
  Nmin is N - 1,
  member(E,L),
  draw(Nmin,L,R).
```

```
confused([]).
confused([_X]).
confused([A,B|C]) :-
  attack_all(A,1,[B|C]),
  confused([B|C]).
attack_all(_,_,[]).
attack_all(A,Off,[B|C]) :-
  Offplus is Off + 1,
  attack(A,Off,B),
  attack_all(A,Offplus,C).
attack(A,_,A).
attack(A,Off,B) :-
  Diff is A - B,
  abs(Diff,Off).
```

Interesting branch

$$\underline{\mathit{draw}(g_1, g_2, a_1)}, \mathit{confused}([g_3|a_1])$$

$$\Big|\quad a_1 = [a_2|a_3]$$

$$\underline{g_1 > 0}, a_4 \text{ is } g_1 - 1, \mathit{member}(a_2, g_2),$$

$$\overline{\mathit{draw}(a}_4, g_2, a_3), \mathit{confused}([g_3, a_2|a_3])$$

$$\vdots\ \cdots$$

$$\mathit{draw}(g_4, g_2, a_3), \mathit{attack\_all}(g_3, g_6, [g_5|a_3]),$$

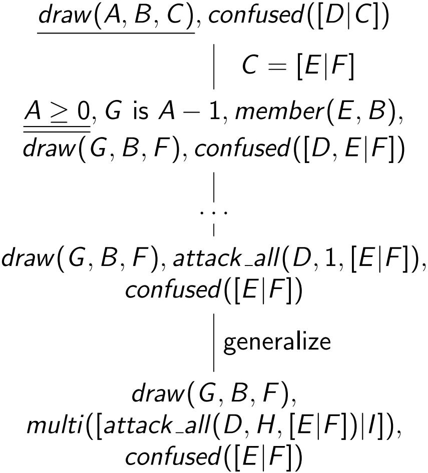
$$\mathit{confused}([g_5|a_3])$$

$$\Big|\ \text{generalize}$$

$$\mathit{draw}(g_4, g_2, a_3), \mathit{multi}(\mathit{attack\_all}(g_3, g_6, [g_5|a_3])),$$

$$\mathit{confused}([g_5|a_3])$$

Interesting branch, synthesis



$$\underline{draw(A,B,C)}, confused([D|C])$$

$$C = [E|F]$$

$$\underline{\underline{A \geq 0}}, G \text{ is } A-1, member(E,B),$$
$$draw(G,B,F), confused([D,E|F])$$

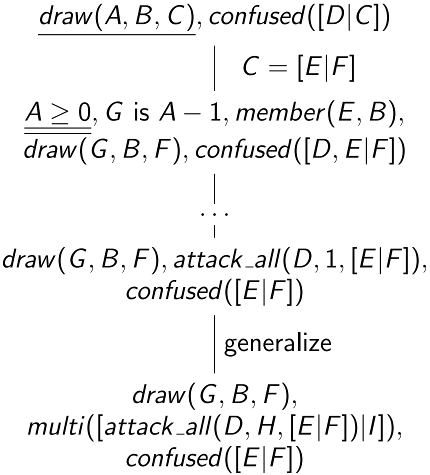
$$\dots$$

$$draw(G,B,F), attack\_all(D,1,[E|F]),$$
$$confused([E|F])$$

generalize

$$draw(G,B,F),$$
$$multi([attack\_all(D,H,[E|F])|I]),$$
$$confused([E|F])$$

Interesting branch, synthesis



$$\underline{\mathit{draw}(A,B,C)}, \mathit{confused}([D|C])$$

$$C = [E|F]$$

$$\underline{\underline{A \geq 0}}, G \text{ is } A-1, \mathit{member}(E,B),$$
$$\mathit{draw}(G,B,F), \mathit{confused}([D,E|F])$$

$$\cdots$$

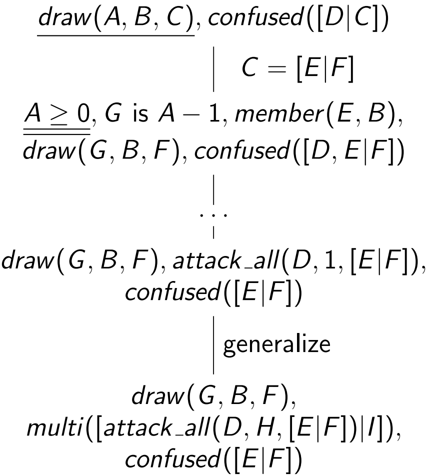
$$\mathit{draw}(G,B,F), \mathit{attack\_all}(D,1,[E|F]),$$
$$\mathit{confused}([E|F])$$

generalize

$$\mathit{draw}(G,B,F),$$
$$\mathit{multi}([\mathit{attack\_all}(D,H,[E|F])|I]),$$
$$\mathit{confused}([E|F])$$

```
draw(A,B,C),
confused([D|C]) <=> C =
[E|F], A>0, G is A-1,
member(E,B), draw(G,B,F),
attack_all(D,1,[E|F]),
confused([E|F]).
```

Interesting branch, synthesis



$\underline{draw(A, B, C)}, confused([D|C])$

$C = [E|F]$

$\underline{\underline{A \geq 0}}, G \text{ is } A - 1, member(E, B),$
$\overline{draw}(G, B, F), confused([D, E|F])$

$\ldots$

$draw(G, B, F), attack\_all(D, 1, [E|F]),$
$confused([E|F])$

generalize

$draw(G, B, F),$
$multi([attack\_all(D, H, [E|F])|I]),$
$confused([E|F])$

```
draw(A,B,C),
confused([D|C]), lock <=>
C = [E|F], A>0, G is A-1,
member(E,B), draw(G,B,F),
attack_all(D,1,[E|F]),
confused([E|F]), lock.
```

Interesting branch, synthesis

$$\underline{draw(A, B, C)}, confused([D|C])$$

$C = [E|F]$

$$\underline{\underline{A \geq 0}}, G \text{ is } A - 1, member(E, B),$$
$$draw(G, B, F), confused([D, E|F])$$

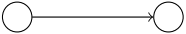
$$\ldots$$

$$draw(G, B, F), attack\_all(D, 1, [E|F]),$$
$$confused([E|F])$$

generalize

$$draw(G, B, F),$$
$$multi([attack\_all(D, H, [E|F])|I]),$$
$$confused([E|F])$$

```
draw(A,B,C),
confused([D|C]), lock <=>
C = [E|F], A>0, G is A-1,
member(E,B), draw(G,B,F),
attack_all(D,1,[E|F]),
confused([E|F]), lock.
```



Interesting branch, synthesis

$$\underline{draw(A, B, C)}, confused([D|C])$$

$$C = [E|F]$$

$$\underline{\underline{A \geq 0}}, G \text{ is } A - 1, member(E, B),$$
$$draw(G, B, F), confused([D, E|F])$$

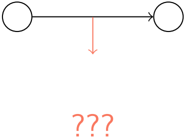
$$\cdots$$

$$draw(G, B, F), attack\_all(D, 1, [E|F]),$$
$$confused([E|F])$$

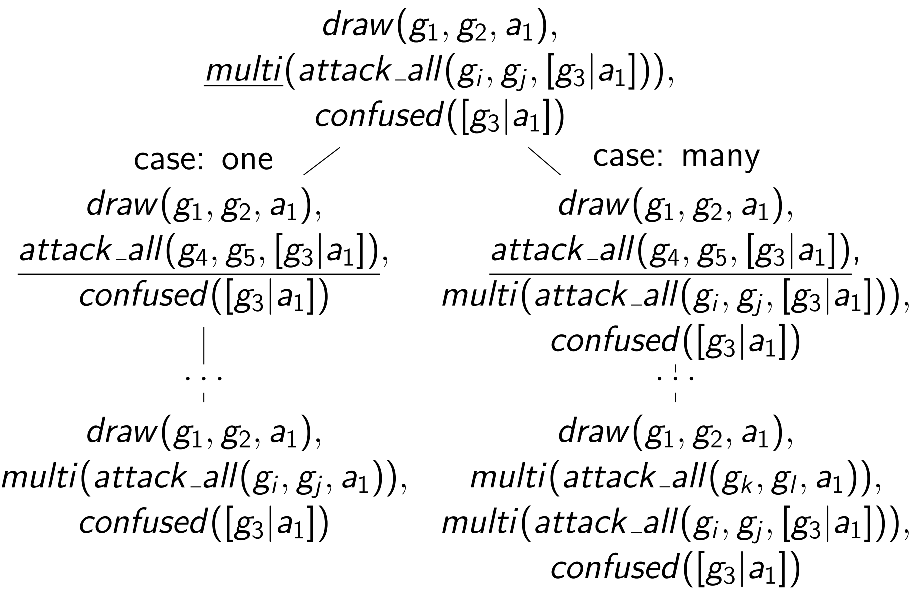
generalize

$$draw(G, B, F),$$
$$multi([attack\_all(D, H, [E|F])|I]),$$
$$confused([E|F])$$

```
draw(A,B,C),
confused([D|C]), lock <=>
C = [E|F], A>0, G is A-1,
member(E,B), draw(G,B,F),
attack_all(D,1,[E|F]),
confused([E|F]), lock.
```



Branches with identical computations



$draw(g_1, g_2, a_1),$
$\underline{multi}(attack\_all(g_i, g_j, [g_3|a_1])),$
$confused([g_3|a_1])$

case: one

$draw(g_1, g_2, a_1),$
$\underline{attack\_all(g_4, g_5, [g_3|a_1])},$
$confused([g_3|a_1])$

$\cdots$

$draw(g_1, g_2, a_1),$
$multi(attack\_all(g_i, g_j, a_1)),$
$confused([g_3|a_1])$

case: many

$draw(g_1, g_2, a_1),$
$\underline{attack\_all(g_4, g_5, [g_3|a_1])},$
$multi(attack\_all(g_i, g_j, [g_3|a_1])),$
$confused([g_3|a_1])$

$\cdots$

$draw(g_1, g_2, a_1),$
$multi(attack\_all(g_k, g_l, a_1)),$
$multi(attack\_all(g_i, g_j, [g_3|a_1])),$
$confused([g_3|a_1])$

Soundness?

```
?- cqueens(4,X).
```

Soundness?

```
?- cqueens(4,X).
X = [1, 1, 1, 1] ;
```

Soundness?

```
?- cqueens(4,X).
X = [1, 1, 1, 1] ;
X = [1, 2, 3, 4] ;
```

Soundness?

```
?- cqueens(4,X).
X = [1, 1, 1, 1] ;
X = [1, 2, 3, 4] ;
X = [2, 2, 2, 2] ;
```

Soundness?

```
?- cqueens(4,X).
X = [1, 1, 1, 1] ;
X = [1, 2, 3, 4] ;
X = [2, 2, 2, 2] ;
X = [3, 3, 3, 3] ;
```

Soundness?

```
?- cqueens(4,X).
X = [1, 1, 1, 1] ;
X = [1, 2, 3, 4] ;
X = [2, 2, 2, 2] ;
X = [3, 3, 3, 3] ;
X = [4, 3, 2, 1] ;
```

Soundness?

```
?- cqueens(4,X).
X = [1, 1, 1, 1] ;
X = [1, 2, 3, 4] ;
X = [2, 2, 2, 2] ;
X = [3, 3, 3, 3] ;
X = [4, 3, 2, 1] ;
X = [4, 4, 4, 4]
```

Soundness?

```
?- cqueens(4,X).
X = [1, 1, 1, 1] ;
X = [1, 2, 3, 4] ;
X = [2, 2, 2, 2] ;
X = [3, 3, 3, 3] ;
X = [4, 3, 2, 1] ;
X = [4, 4, 4, 4]
```

```
?- cqueens(4,[A,B,C,D]).
```

Soundness?

```
?- cqueens(4,X).
X = [1, 1, 1, 1] ;
X = [1, 2, 3, 4] ;
X = [2, 2, 2, 2] ;
X = [3, 3, 3, 3] ;
X = [4, 3, 2, 1] ;
X = [4, 4, 4, 4]
```

```
?- cqueens(4,[A,B,C,D]).
A=B, B=C, C=D, D=2 ;
```

Soundness?

```
?- cqueens(4,X).
X = [1, 1, 1, 1] ;
X = [1, 2, 3, 4] ;
X = [2, 2, 2, 2] ;
X = [3, 3, 3, 3] ;
X = [4, 3, 2, 1] ;
X = [4, 4, 4, 4]
```

```
?- cqueens(4,[A,B,C,D]).
A=B, B=C, C=D, D=2 ;
ERROR: is/2:  Arguments
are not sufficiently
instantiated
```

The culprit

$$\begin{gathered}
draw(g_1, g_2, a_1), \underline{multi}(attack\_all(g_i, g_j, [g_3|a_1])), \\
confused([g_3|a_1]) \\
\Big| \\
\cdots \\
\Big| \\
draw(g_1, g_2, a_1), multi(attack\_all(g_i, g_j, a_1)), \\
confused([g_3|a_1])
\end{gathered}$$

```
lock, attack_all(D,E,[F|C]) <=>
H is E + 1, attack(D,E,F),
attack_all(D,H,C), lock.
```

The culprit

$$\mathit{draw}(g_1, g_2, a_1), \underline{\mathit{multi}}(\mathit{attack\_all}(g_i, g_j, [g_3|a_1])),$$
$$\mathit{confused}([g_3|a_1])$$

$$\Big|$$

$$\ldots$$

$$\Big|$$

$$\mathit{draw}(g_1, g_2, a_1), \mathit{multi}(\mathit{attack\_all}(g_i, g_j, a_1)),$$
$$\mathit{confused}([g_3|a_1])$$

```
lock, attack_all(D,E,[F|C]) <=>
H is E + 1, attack(D,E,F),
attack_all(D,H,C), lock.
```

The solution

- encode *assumed* level of instantiation in constraints

The solution

- encode *assumed* level of instantiation in constraints
- atomically renaming conjunctions (in Prolog) does this *implicitly*

The solution

- encode *assumed* level of instantiation in constraints
- atomically renaming conjunctions (in Prolog) does this *implicitly*

```
lock, attack_all(D,E,[F|C]) <=> H is E + 1,
attack(D,E,F), attack_all(D,H,C), lock.
```

The solution

- encode *assumed* level of instantiation in constraints
- atomically renaming conjunctions (in Prolog) does this *implicitly*

```
lock, attack_all(D,E,[F|C],[g,g,[g|a]]) <=>
H is E + 1, attack(D,E,F),
attack_all(D,H,C,[g,g,a]), lock.
```

Code (Prolog)

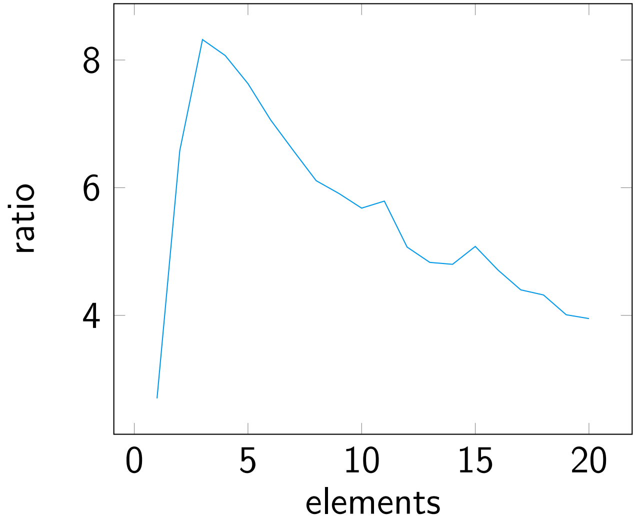
```
genlist(N,L) :- N >= 1, genlist_acc(N,[],L).
genlist_acc(N,Acc,L) :- N > 1, Nmin is N-1, genlist_acc(Nmin,[N|Acc],L).
genlist_acc(N,Acc,[1|Acc]) :- N is 1.
attack(A,_,A).
attack(A,Offset,B) :-
  Diff is A - B,
  abs(Diff,Offset).
queens(0,[]).
queens(A,[D|E]) :-
  genlist(A,C),
  A > 0,
  F is A - 1,
  member(D,C),
  a(draw(F,C,E),confused([D|E])).
a(draw(0,_,[]),confused([_])).
a(draw(A,B,[E|F]),confused([D,E|F])) :-
  A >= 0,
  G is A - 1,
  member(E,B),
  b(draw(G,B,F),
    multi([attack_all(D,1,[E|F])]),
    confused([E|F])).
b(draw(A,B,C),multi([attack_all(D,E,[F|C])]),confused([F|C])) :-
  H is E + 1,
  attack(D,E,F),
  c(draw(A,B,C),
    multi([attack_all(D,H,C)]),
    confused([F|C])).
b(draw(A,B,C),multi([attack_all(D,E,[F|C]),
  attack_all(H,I,[F|C])|J]),confused([F|C])) :-
  K is E + 1,
  attack(D,E,F),
  d(draw(A,B,C),multi([attack_all(D,K,C)]),
    multi([attack_all(H,I,[F|C])|J]),confused([F|C])).
d(draw(A,B,C),multi([attack_all(D,E,C)|F]),
  multi([attack_all(G,H,[I|C])]),confused([I|C])) :-
  K is H + 1,
  attack(G,H,I),
  append([attack_all(D,E,C)|F],[attack_all(G,K,C)],
         Appended),
  c(draw(A,B,C),multi(Appended),confused([I|C])).
```

```
d(draw(A,B,C),multi([attack_all(D,E,C)|F]),
  multi([attack_all(G,H,[I|C]),attack_all(K,L,[I|C])|M]),
  confused([I|C])) :-
  N is H + 1,
  attack(G,H,I),
  append([attack_all(D,E,C)|F],[attack_all(G,N,C)],
         Appended),
  d(draw(A,B,C),multi(Appended),
    multi([attack_all(K,L,[I|C])|M]),confused([I|C])).
c(draw(0,B,[]),multi([attack_all(D,E,[])|F]),confused([G])) :-
  e(multi([attack_all(D,E,[])|F]),confused([G])).
c(draw(A,B,[H|I]),multi([attack_all(D,E,[H|I])|F]),
  confused([G,H|I])) :-
  A > 0,
  J is A - 1,
  member(H,B),
  append([attack_all(D,E,[H|I])|F],
         [attack_all(G,1,[H|I])],Appended),
  b(draw(J,B,I),multi(Appended),confused([H|I])).
e(multi([attack_all(A,B,[])]),confused([Z])).
e(multi([attack_all(A,B,[]),
  attack_all(C,D,[])|E]),confused([Z])) :-
  e(multi([attack_all(C,D,[])|E]),confused([Z])).
```

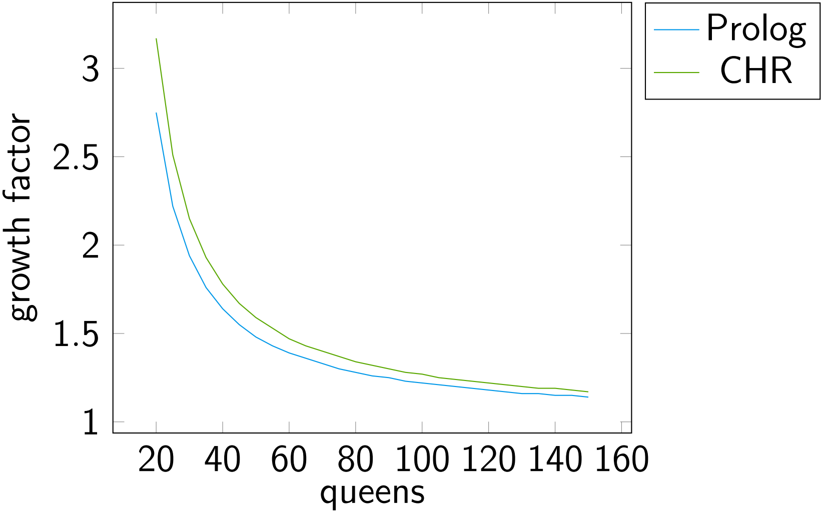
Code (CHR)

```
genlist(N,L) :- N >= 1, genlist_acc(N,[],L).
genlist_acc(N,Acc,L) :- N > 1, Nmin is N-1, genlist_acc(Nmin,[N|Acc],L).
genlist_acc(N,Acc,[1|Acc]) :- N is 1.
attack(A,_,A).
attack(A,Offset,B) :- Diff is A - B, abs(Diff,Offset).
lock, rename, attack_all(A,B,C,[g,g,a]) <=>
  attack_all(A,B,C,[g,g,[g|a]]), rename, lock.
lock, rename <=> lock.
cqueens(0,B,[g,a]) <=> genlist(0,C), B = [].
cqueens(A,B,[g,a]) <=>
  B = [E|F],
  genlist(A,C),
  A > 0,
  D is A-1,
  member(E,C),
  draw(D,C,F,[g,g,a]),
  confused([E|F],[g|a]),
  lock.
attack_all(A,Off,[B|C],[g,g,[g|a]]), lock <=>
  Off1 is Off + 1, attack(A,Off,B), attack_all(A,Off1,C,[g,g,a]), lock.
attack_all(X,Y,[],_), lock <=> lock.
draw(0,B,C,[g,g,a]), confused([D|C],[g|a]), lock <=> C = [], lock.
draw(A,B,C,[g,g,a]), confused([D|C],[g|a]), lock <=>
  C = [E|F],
  A>0,
  G is A-1,
  member(E,B),
  draw(G,B,F,[g,g,a]),
  rename,
  attack_all(D,1,[E|F],[g,g,[g|a]]),
  confused([E|F],[g|a]),
  lock.
lock <=> true.
```

Permutation sort: $\frac{\text{inferences CHR}}{\text{inferences Prolog}}$



Queens: $\frac{\text{inferences } \mathit{cqueens}(N,X)}{\text{inferences } \mathit{cqueens}(N-1,X)}$



Results

- Concise representation

Results

- Concise representation
- Extra target for analysis

Results

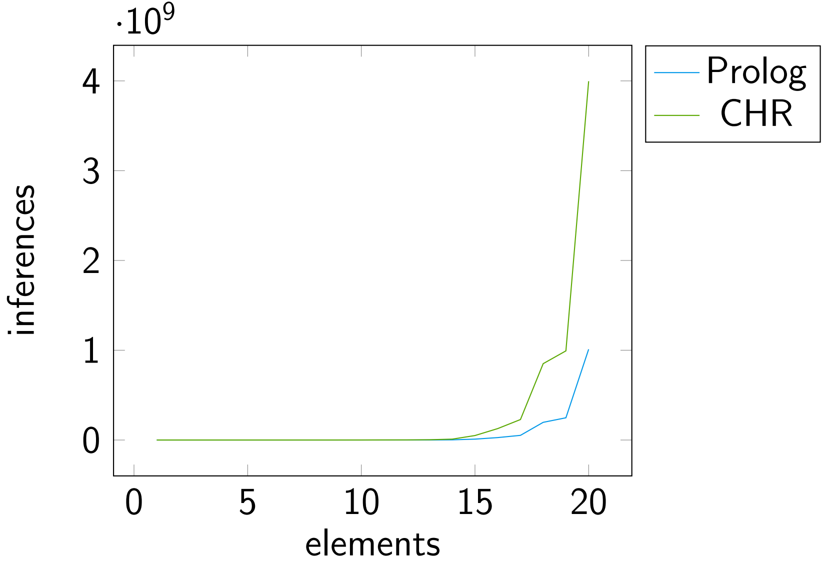
- Concise representation
- Extra target for analysis
- Portability to other CHR systems

Results

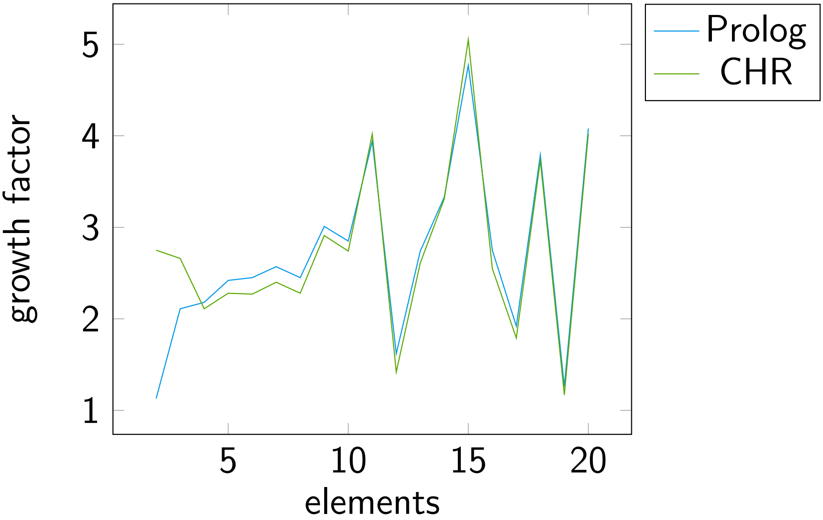
- Concise representation
- Extra target for analysis
- Portability to other CHR systems
- Slower resulting programs, but potentially same asymptotic time complexity

Questions?

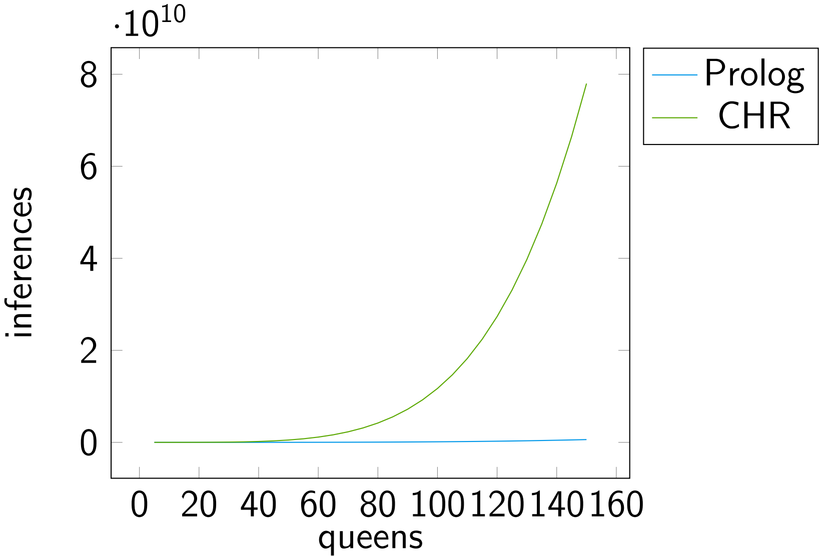
Permutation sort: inferences



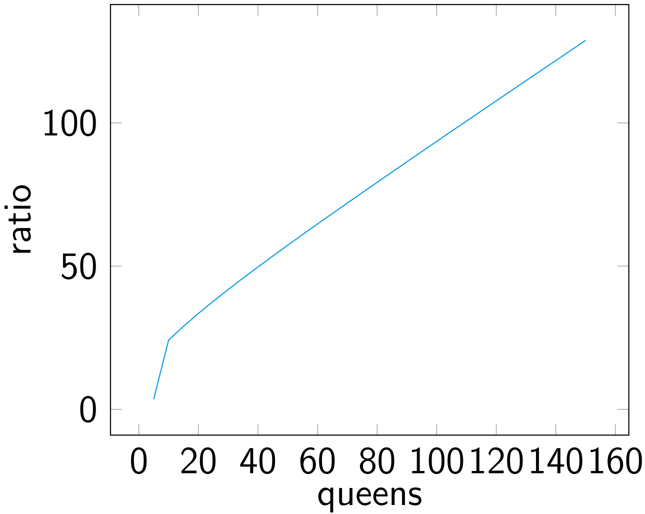
Permutation sort: growth w.r.t. previous size



Queens: inferences



Queens: $\frac{\text{inferences CHR}}{\text{inferences Prolog}}$


ratio
queens