Practice of Answer Set Programming

Methodology of ASP

Ira A. Fulton Schools of Engineering
Arizona State University

Objectives

Objective

Use Generate-(Define)-Test method of organizing ASP rules

Queens Puzzle

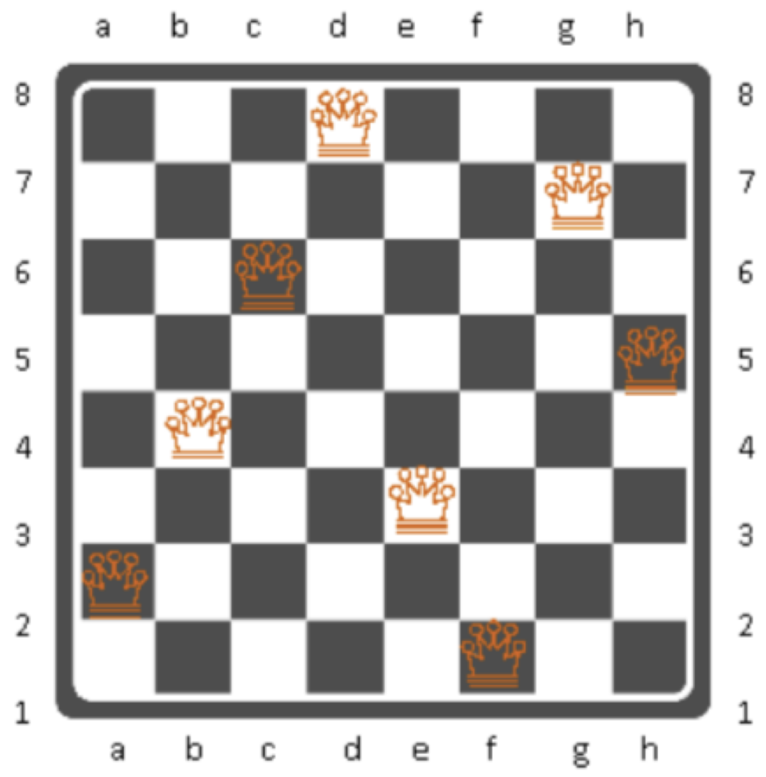
Generate-(Define)-Test

A way to organize rules in ASP

- GENERATE part: generates a “search space” – a set of potential solutions
- DEFINE part: defines new atoms in terms of other atoms
- TEST part: weed out the elements of the search space that do not represent solutions

N-Queens Puzzle

No two queens can share the same row, column, or diagonal.

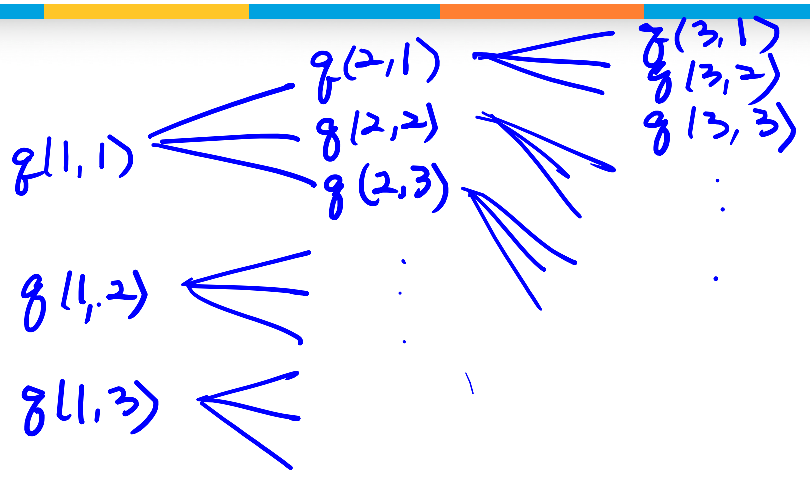


n	# sol
3	none
4	2
5	10
6	4
7	40
8	92

N-Queens in ASP

```
% Each row has exactly one queen
1 {queen(R,1..n)} 1 :- R=1..n.

% or
```

{queen(R,1..$\overset{=3}{n}$)}=1 :- R=1..$\overset{=3}{n}$.



$\Rightarrow$ {q(1, 1..3)} = 1 (R=1) $\Rightarrow$ {q(1, 1); q(1,2); q(1,3)} = 1

{q(2, 1..3)} = 1 (R=2) {q(2,1); q(2,2); q(2,3)} = 1

{q(3, 1..3)} = 1 (R=3) {q(3,1); q(3,2); q(3,3)} = 1

N-Queens in ASP

```
% Each row has exactly one queen
{queen(R,1..n)}=1 :- R=1..n.


% No two queens are on the same column
:- queen(R1,C), queen(R2,C), R1!=R2.


% No two queens are on the same diagonal
:- queen(R1,C1), queen(R2,C2), R1!=R2, |R1-R2|=|C1-C2|.
          1. 1          2  2
```

Finding One Solution for the 8-Queens Puzzle

```
$ clingo queens.lp -c n=8
clingo version 5.2.1
Reading from queens.lp
Solving...
Answer: 1
queen(4,1) queen(6,2) queen(8,3) queen(2,4) queen(7,5) queen(1,6)
queen(3,7) queen(5,8)
SATISFIABLE

Models       : 1+
Calls        : 1
Time         : 0.004s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.004s
```

Finding All Solutions for the 8-Queens Puzzle

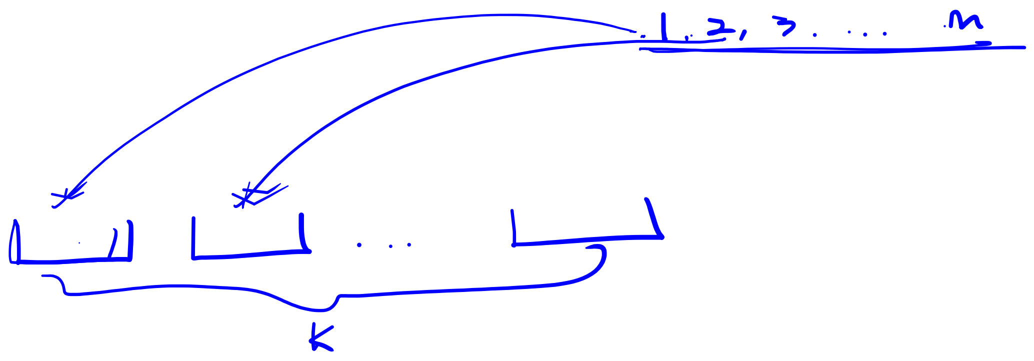
```
$ clingo queens.lp -c n=8 0
clingo version 5.2.1
Reading from queens.lp
Solving...
Answer: 1
queen(4,1) queen(6,2) queen(8,3) queen(2,4) queen(7,5) queen(1,6)
queen(3,7) queen(5,8)
Answer: 2
[[ truncated ]]
Answer: 92
queen(5,1) queen(1,2) queen(8,3) queen(4,4) queen(2,5) queen(7,6)
queen(3,7) queen(6,8)
SATISFIABLE

Models       : 92
Calls        : 1
Time         : 0.011s (Solving: 0.01s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.010s
```

Schur Numbers

Schur Numbers

- A set A of integers is called **sum-free** if there are no numbers x, y in A such that x+y is in A also
- The Schur number S(k) is the largest integer n for which the interval {1, . . . , n} can be partitioned into k sum-free sets
- S(1) = 1
- S(2) = 4
- S(3) = 13

{1, 2} {1, 3}



1

1 3 | 2 | (X) | 1 4 | 2 . 3

1, 4. 7, 10, 13 | 2, 3, 11, 12 | 5, 6, 8. 9

Schur Numbers

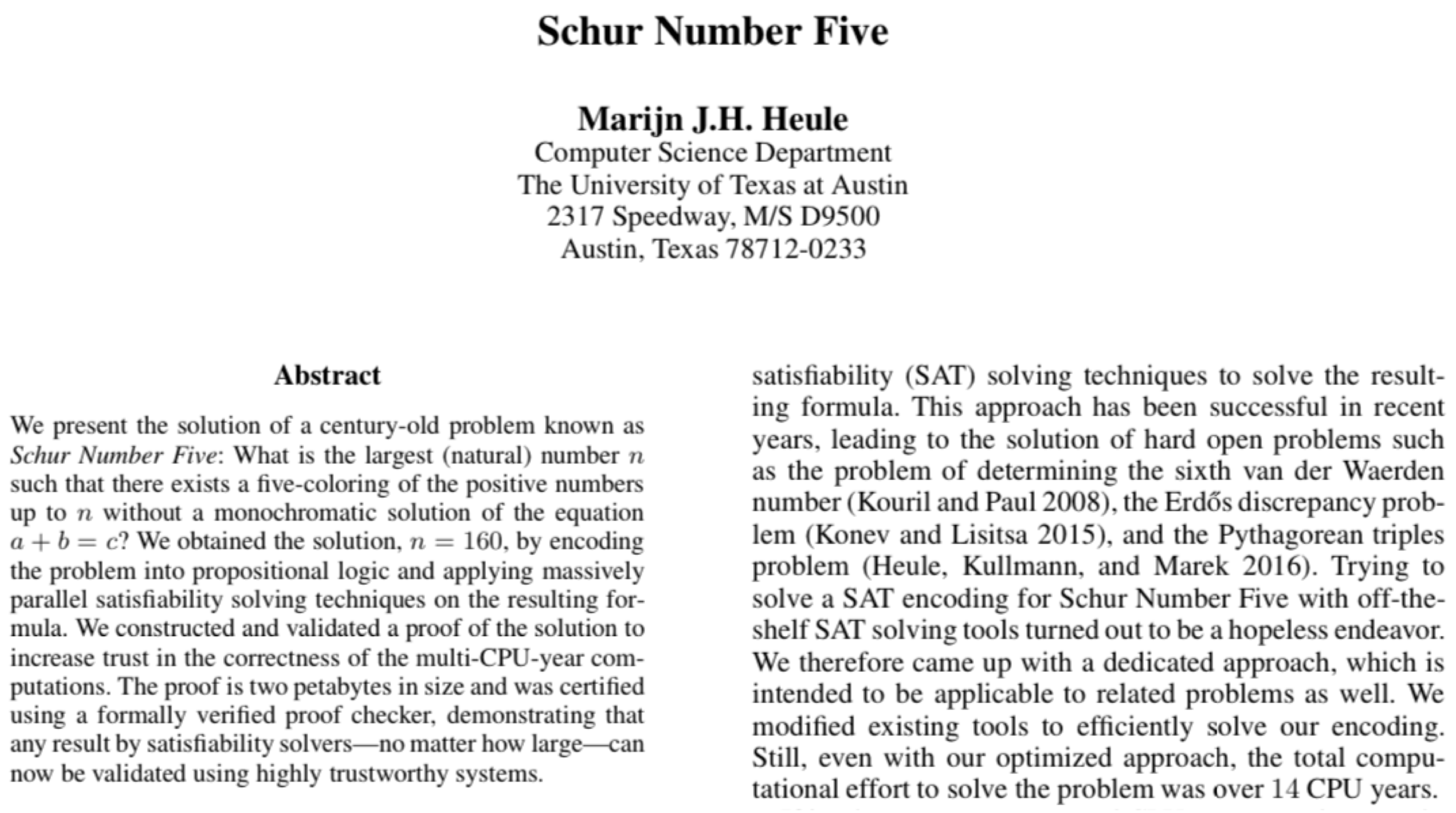
- A set A of integers is called **sum-free** if there are no numbers x, y in A such that x+y is in A also
- The Schur number S(k) is the largest integer n for which the interval {1, . . . , n} can be partitioned into k sum-free sets
- S(1) =
- S(2) =
- S(3) =

S(k)	
S(1)=1	
S(2)=4	
S(3)=13	
S(4)=44	(Baumert 1965, Abbott and Hanson 1972)
160<=S(5)<=315 S(5) = 160	(Fredriksen 1979) (Heule 2017) https://arxiv.org/pdf/1711.08076.pdf
S(6) >= 536	
S(7) >= 1680	(Fredriksen and Sweet 2000)
S(n) <= R(n)-2	

Schur Numbers in ASP

```
% Partition {1,..,n} into k sum-free sets

% input: positive integers n, k.

% partition set {1,...,n} into k subsets
% in(I,S) means number I is in set S
{in(I,1..k)} = 1 :- I=1..n.

% these subsets are sum-free.
:- in(I,S), in(J,S), in(I+J,S).

Command: clingo –c k=3 -c n=12 schur.lp [–t4]
```

Finding S(5)=160



Schur Number Five

Marijn J.H. Heule
Computer Science Department
The University of Texas at Austin
2317 Speedway, M/S D9500
Austin, Texas 78712-0233

Abstract

We present the solution of a century-old problem known as *Schur Number Five*: What is the largest (natural) number n such that there exists a five-coloring of the positive numbers up to n without a monochromatic solution of the equation $a + b = c$? We obtained the solution, $n = 160$, by encoding the problem into propositional logic and applying massively parallel satisfiability solving techniques on the resulting formula. We constructed and validated a proof of the solution to increase trust in the correctness of the multi-CPU-year computations. The proof is two petabytes in size and was certified using a formally verified proof checker, demonstrating that any result by satisfiability solvers—no matter how large—can now be validated using highly trustworthy systems.

satisfiability (SAT) solving techniques to solve the resulting formula. This approach has been successful in recent years, leading to the solution of hard open problems such as the problem of determining the sixth van der Waerden number (Kouril and Paul 2008), the Erdős discrepancy problem (Konev and Lisitsa 2015), and the Pythagorean triples problem (Heule, Kullmann, and Marek 2016). Trying to solve a SAT encoding for Schur Number Five with off-the-shelf SAT solving tools turned out to be a hopeless endeavor. We therefore came up with a dedicated approach, which is intended to be applicable to related problems as well. We modified existing tools to efficiently solve our encoding. Still, even with our optimized approach, the total computational effort to solve the problem was over 14 CPU years.

Wrap-Up