

数独专家系统实验报告

14307130078 张博洋

一、概述

实现了一套用于解数独谜题的专家系统，其中包含 18 条规则（由网上搜集的数独技巧归纳而来），经测试可以解出较难难度的谜题。

整个专家系统全部是自己编写（规则格式和推理机也是自己编写的），编程语言采用 JavaScript。

二、使用方法

打开 sudoku.html 即可使用。整个界面由很多个文本框组成，它们的含义如下：

| | |
|----------------|-------------------------------|
| RULES（规则） | 专家系统中用到的规则，可以根据需要进行修改 |
| PUZZLE（谜题） | 在此区域接受用户输入 |
| ANSWER（解答） | 运行完毕后会在此显示解题结果 |
| LOG（日志） | 运行完毕后会在此显示推理过程和中间事实 |
| QUERY（查询） | 在运行完毕后，可以在此查询某一条具体事实的推理过程 |
| COMPILED RULES | 在此显示已编译（转换为 JavaScript 代码）的规则 |

如要解谜题，只要在 PUZZLE 文本框中输入数独题面（以 0 代替空位，已经有一个样例题面预先填好），然后点击文本框下的 SOLVE 按钮即可。一般来说运行时间不会超过 10 秒钟。

三、实现

1. 数独简介

| | 1列 | 2列 | 3列 | 4列 | 5列 | 6列 | 7列 | 8列 | 9列 |
|----|----|----|----|----|----|----|----|----|----|
| 1行 | | | | | | | | | |
| 2行 | | 1宫 | | | 2宫 | | | 3宫 | |
| 3行 | | | | | | | | | |
| 4行 | | | | | | | | | |
| 5行 | | 4宫 | | | 5宫 | | | 6宫 | |
| 6行 | | | | | | | | | |
| 7行 | | | | | | | | | |
| 8行 | | 7宫 | | | 8宫 | | | 9宫 | |
| 9行 | | | | | | | | | |

数独游戏的目标是在已经填入一部分数字的 9×9 格子中，使用推理等方法填入剩下的数字，使得每一行、每一列、每一宫中 1~9 这 9 个数字各出现且仅出现一次。一般认为数独题面是有唯一解的。

2. 事实

由于数独游戏规则较为简单，因此设计了两种事实类型：

- (1) 某位置 (i, j) 一定是某数 (n)：用 mustbe(i, j, n) 表示
- (2) 某位置 (i, j) 一定不是某数 (n)：用 cantbe(i, j, n) 表示

mustbe() 和 cantbe() 取值只能是 true 或 false 之一，表示对应事实是否成立，如果不确定则取值 false。

此外为了规则表示方便，还设计了几个辅助函数：

- (1) $\text{sure}(i, j)$ 表示某位置是否已经确定（即是否存在 n 使得 $\text{mustbe}(i, j, n)$ 的值为 true）
- (2) $\text{a}(i, j)$ ：当某位置对应数字已经确定时， $\text{a}(i, j)$ 返回这个数字；否则无意义
- (3) $\text{gong}(i, j)$ ：计算位置 (i, j) 对应的宫的编号

3. 规则

专家系统内嵌了 18 条规则，分为两类：简单规则和高级规则。简单规则计算速度快也较为常用；高级规则可以解决简单规则不能解决的情况，但是计算速度较慢。

规则的表达使用了自己定义的格式，在专家系统运行时会将规则文本转换为代码执行。由于数独的特点，规则中所有变量的取值范围都是 1~9。

(1) 简单规则

简单规则一共 6 条，已经足以对付常见的数独谜题了。

- a. 规则 1：若某个位置只有一种可能的数，则此位置一定是此数

规则代码：

```
RULE /* 1 */
FORALL{num, i, j} /* 枚举数 num, 位置(i,j) */
/* 如果对任意 k (k != num) 都成立 cantbe(i, j, k) */
IF ANY{k: k != num: cantbe(i, j, k)}
/* 则可以导出事实 mustbe(i, j, num) */
THEN export_mustbe(i, j, num)
```

- b. 规则 2：若一行中其他位置都不能为某数，则此位置一定是此数

- c. 规则 3：若一列中其他位置都不能为某数，则此位置一定是此数

- d. 规则 4：若某宫中其他位置都不能为某数，则此位置一定是此数

规则代码（仅以行为例）：

```
RULE /* 2 */
FORALL{num, i, j} /* 枚举数 num, 位置(i,j) */
/* 如果对任意 k (k != j) 都成立 cantbe(i, k, num) */
IF ANY{k: k != j: cantbe(i, k, num)}
/* 则可以导出事实 mustbe(i, j, num) */
THEN export_mustbe(i, j, num)
```

- e. 规则 5：若已确定某一位置是某数，则同一行、列、宫的其他位置都不能是此数

规则代码：

```
RULE /* 5 */
FORALL{i, j, x, y} /* 枚举位置(i,j), 位置(x,y) */
/* 如果(i,j)取值已确定，且(i,j)和(x,y)在同一行或列或宫 */
IF sure(i, j) && (i != x || j != y) && (i == x || j == y ||
gong(i, j) == gong(x, y))
/* 则(x,y)不能取值 a(i,j)，即可以导出事实 cantbe(x,y, a(i,j)) */
THEN export_cantbe(x, y, a(i, j))
```

f. 规则 6: 若已确定某一位置是某数, 则此位置不能是其他数
规则代码:

```
RULE /* 6 */
FORALL{i, j, num} /* 枚举数 num, 位置(i,j) */
/* 如果(i,j)取值已确定, 且取值不是 num */
IF sure(i, j) && a(i, j) != num
/* 则可以导出事实 cantbe(i, j, num) */
THEN export_cantbe(i, j, num)
```

(2) 高级规则

高级规则一共 12 条, 可以对付较难的数独谜题。

a. 规则 7,8,9,10: 区块摒除法

| | 1列 | 2列 | 3列 | 4列 | 5列 | 6列 | 7列 | 8列 | 9列 |
|----|----|----|----|----|----|----|----|----|----|
| 1行 | | | | | | | | | |
| 2行 | | | | | | | | | |
| 3行 | | | | | | | | | |

区块摒除法的思想是: 如果一行(或列)与一个宫相交(设如上图所示), 且行中不与宫相交的部分(黄色)都不能取某数 n , 则说明行与宫相交的部分(红色)中一定有一个位置取 n , 因此宫中不与行相交的部分(绿色)都一定不能取 n 。

这种思想对于列也成立。此外, 也可以根据宫的信息推出行(或列)的信息。因此区块摒除法的规则一共有 4 条。

规则代码(以由行推宫为例):

```
RULE ADVANCED /* 7 */
FORALL{num, i, j} /* 枚举数 num, 行号 i, 宫号 j */
/* 如果行 i 和宫 j 相交; 并且对于在行 i 中而不在宫 j 中的格子(i,k)都成立 cantbe(i, k, num) */
IF EXISTS{k: true: gong(i, k) == j} && ANY{k: gong(i, k) != j: cantbe(i, k, num)}
/* 则对于在行 i 和宫 j 相交的格子(x,y)都可以导出 cantbe(x,y,num) */
THEN ALL{x, y: gong(x, y) == j && x != i: export_cantbe(x, y, num)}
```

b. 规则 11,12: X-WING 法

| | 1列 | 2列 | 3列 | 4列 | 5列 | 6列 | 7列 | 8列 | 9列 |
|----|----|----|----|----|----|----|----|----|----|
| 1行 | | | | | | | | | |
| 2行 | | | | | | | | | |
| 3行 | | | | | | | | | |
| 4行 | | | | | | | | | |
| 5行 | | | | | | | | | |
| 6行 | | | | | | | | | |
| 7行 | | | | | | | | | |
| 8行 | | | | | | | | | |
| 9行 | | | | | | | | | |

X-WING 的思想是：如果某两行和某两列中（设如上图所示），行中不与列相交的部分（绿色）都不能取某数 n ，则说明行与列相交的部分（红色）中要么是左上右下取 n 要么是左下右上取 n ，因此列中的两个相交的位置（某一列中红色）一定有一个位置取 n ，因此列中不与行相交的部分（黄色）都一定不能取 n 。

这种思想也可以由列推行。因此 X-WING 的规则一共有 2 条。

规则代码（以由行推列为例）：

```
RULE ADVANCED /* 11 */
FOR{a, b: a != b} /* 枚举两行 */
FOR{c, d: c != d} /* 枚举两列 */
FORALL{num} /* 枚举数 num */
/* 如果行 a 和行 b 中不与列 c 列 d 相交的部分都不能取 num */
IF ANY{k: k != c && k != d: cantbe(a, k, num) && cantbe(b, k, num)}
/* 则列 c 和列 d 中不与行 a 行 b 相交的部分也不能取 num */
THEN ALL{k: k != a && k != b: export_cantbe(k, c, num), export_cantbe(k, d, num)}
```

c. 规则 13,14,15,16,17,18: 数对法

| | 1列 | 2列 | 3列 | 4列 | 5列 | 6列 | 7列 | 8列 | 9列 |
|----|----|----|----|----|----|----|----|----|----|
| 1行 | | | | | | | | | |

数对法的思想是：如果某行中除某两个位置（红色）外的其他位置（黄色）都不能取某两个数 n 、 m ，则说明这 n 、 m 一定在两个位置（红色）中，因此这两个位置（红色）不能取除 n 、 m 以外的任何数。

这种思想也可以应用到列、宫上。此外还可以反推（即某两位置只能取某两数时，其他位置就不能取这两数了）。因此数对法的规则一共有 6 条。

规则代码（以由行推数对为例）：

```
RULE ADVANCED /* 11 */
FOR{a, b: a != b} /* 枚举两数 a,b */
FOR{s, t: s != t} /* 枚举两列 s,t */
FORALL{i} /* 枚举行 i */
/* 如果 i 中不与列 s、t 相交的部分都不能取 a 和 b */
IF ANY{k: k != s && k != t: cantbe(i, k, a) && cantbe(i, k, b)}
/* 则相交部分不能取 a、b 以外的任何数 */
THEN ALL{z: z != a && z != b: export_cantbe(i, s, z), export_cantbe(i, t, z)}
```

4. 推理机

推理机采用前向链接方式进行推理。每轮迭代中会依次尝试每条规则，如果所有规则都无法推出新的事实，则算法结束。此外有一个小优化：简单规则优先。在一轮迭代中简单规则无法推出新事实时才会尝试高级规则。推理过程中还会记录每条事实的推理过程以便后续查询。

四、使用示例

1. 打开 `sudoku.html`，PUZZLE 文本框中已填好样例题面（如图），点击 SOLVE 按钮开始求解

PUZZLE: (input your puzzle here, use 0 for blank positions)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 6 | 0 | 5 | 1 | 0 | 3 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 3 |
| 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 |
| 0 | 8 | 6 | 0 | 1 | 0 | 0 | 0 | 5 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 |
| 1 | 0 | 0 | 0 | 2 | 0 | 3 | 4 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 7 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 9 | 0 | 4 | 8 | 0 | 1 |

2. 求解完毕后，ANSWER 文本框中会显示出求解结果，LOG 文本框中会显示日志

ANSWER:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 6 | 4 | 5 | 1 | 9 | 3 | 7 | 8 | 2 |
| 2 | 7 | 1 | 8 | 4 | 5 | 9 | 6 | 3 |
| 9 | 3 | 8 | 6 | 7 | 2 | 1 | 5 | 4 |
| 3 | 8 | 6 | 4 | 1 | 7 | 2 | 9 | 5 |
| 4 | 5 | 2 | 3 | 8 | 9 | 6 | 1 | 7 |
| 1 | 9 | 7 | 5 | 2 | 6 | 3 | 4 | 8 |
| 8 | 6 | 4 | 7 | 3 | 1 | 5 | 2 | 9 |
| 7 | 1 | 9 | 2 | 5 | 8 | 4 | 3 | 6 |
| 5 | 2 | 3 | 9 | 6 | 4 | 8 | 7 | 1 |

LOG:

```
INFO: RULE 17, fires 0 times, finished in 0.07443499999999949s
INFO: RULE 18, fires 0 times, finished in 0.12723999999999977s
INFO: ITERATION 15 takes 0.80202999999999988s
INFO: ===== SUMMARY =====
INFO: FINISHED in 2.5183850000000003s
INFO: 0 UNRESOLVED
INFO: TOTAL 15 ITERATIONS
INFO: RULE SUMMARY:
INFO: RULE 1 fires 37 times, cputime 0.039360000000000422s
INFO: RULE 2 fires 16 times, cputime 0.03656999999999971s
INFO: RULE 3 fires 4 times, cputime 0.030944999999999425s
INFO: RULE 4 fires 0 times, cputime 0.06538999999999942s
INFO: RULE 5 fires 453 times, cputime 0.109145000000000589s
INFO: RULE 6 fires 125 times, cputime 0.021929999999999847s
INFO: RULE 7 fires 5 times, cputime 0.013119999999999891s
INFO: RULE 8 fires 1 times, cputime 0.01374500000000008s
INFO: RULE 9 fires 1 times, cputime 0.0155900000000001965s
INFO: RULE 10 fires 1 times, cputime 0.013899999999999637s
INFO: RULE 11 fires 0 times, cputime 0.21914500000000042s
INFO: RULE 12 fires 0 times, cputime 0.199255000000000285s
INFO: RULE 13 fires 5 times, cputime 0.21364999999999965s
INFO: RULE 14 fires 5 times, cputime 0.207494999999999899s
INFO: RULE 15 fires 3 times, cputime 0.47974500000000008s
INFO: RULE 16 fires 10 times, cputime 0.20964499999999986s
INFO: RULE 17 fires 11 times, cputime 0.20679500000000019s
INFO: RULE 18 fires 4 times, cputime 0.3530049999999992s
```

3. 如果对某条事实的推理过程感兴趣，可以在 QUERY 框中输入编号并按回车，下方文本框中会显示具体信息。下图是查询 572 号事实的推理过程的截图。查询得到的信息包括：推理所用规则（RULE）、推理结论（RESULT）、规则中各变量取值（RULE VARS）、当时数独状态（STATUS）、规则使用哪些规则（USED FACTS）。

五、性能测试

第一个网站是 www.websudoku.com，此网站上谜题难度分为四个等级：Easy, Medium, Hard, Evil。随机选取 5 个 Evil 级别的谜题，都可以解出。

可以得出结论，本专家系统可以解出较难难度的谜题。

1. 《标准数独解题之旅（用一道数独题讲解最基本的 5 种解题技巧）》
<http://www.sudokufans.org.cn/forums/topic/8>
2. <http://www.learn-sudoku.com>
3. http://www.sudokuwiki.org/naked_candidates

七、附：数独谜题样例

| | | | |
|----------|---|---|---|
| 题面 | 2 0 0 0 0 0 5 0 0 0 0 0 2 0 0 9 0 7 0 4 0 5 0 0 0 0 0 0 7 0 0 8 3 0 0 2 0 1 0 0 0 0 0 7 0 5 0 0 7 2 0 0 6 0 0 0 0 0 0 8 0 2 0 3 0 1 0 0 9 0 0 0 0 0 7 0 0 0 0 0 4 | 0 0 0 4 0 2 0 9 0 0 4 0 0 0 0 0 6 0 1 0 0 0 5 0 3 0 0 6 0 8 9 4 0 0 0 0 7 0 0 0 0 0 0 0 1 0 0 0 0 2 3 4 0 6 0 0 1 0 8 0 0 0 2 0 7 0 0 0 0 0 1 0 0 6 0 1 0 7 0 0 0 | 2 0 0 0 0 8 3 0 5 1 0 0 0 0 6 0 0 0 9 6 0 0 7 0 2 0 0 0 5 0 6 0 0 8 0 0 0 0 0 0 2 0 0 0 0 0 0 2 0 0 1 0 9 0 0 0 1 0 6 0 0 5 3 0 0 0 3 0 0 0 0 8 4 0 3 5 0 0 0 0 6 |
| 来源 | websudoku Evil 难度 | sudoku-solutions Medium 难度 | sudoku-solutions Hard 难度 |
| 状态 | 可以解出 | 可以解出 | 不能解出（余 25 空） |
| 求解 结果 | 2 9 3 8 4 7 5 1 6 1 8 5 2 3 6 9 4 7 7 4 6 5 9 1 2 3 8 6 7 9 1 8 3 4 5 2 4 1 2 9 6 5 8 7 3 5 3 8 7 2 4 1 6 9 9 6 4 3 5 8 7 2 1 3 2 1 4 7 9 6 8 5 8 5 7 6 1 2 3 9 4 | 3 5 6 4 7 2 1 9 8 9 4 7 3 1 8 2 6 5 1 8 2 6 5 9 3 4 7 6 2 8 9 4 1 5 7 3 7 3 4 8 6 5 9 2 1 5 1 9 7 2 3 4 8 6 4 9 1 5 8 6 7 3 2 8 7 5 2 3 4 6 1 9 2 6 3 1 9 7 8 5 4 | 2 . . 1 9 8 3 6 5 1 3 8 2 5 6 . . 9 9 6 5 4 7 3 2 8 1 3 5 . 6 4 . 8 1 2 . 1 . . 2 5 6 3 . 6 . 2 . 3 1 5 9 . . . 1 . 6 . . 5 3 5 . 6 3 1 . . . 8 4 . 3 5 8 . 1 . 6 |