Язык *MiniZinc* как средство формулировки задач удовлетворения ограничений

Курс «Введение в решение задач при помощи методов удовлетворения ограничений»

НИУ ИТМО, кафедра «Компьютерные технологии»

Вспомним основные понятия

- Переменная
- Область значений переменной (domain)
- Ограничение
- Задача – найти значения переменных, удовлетворяющие всем ограничениям

Язык *MiniZinc*

- http://www.minizinc.org/
- Поддержка целочисленных и вещественных переменных
- Возможность решения задач минимизации и максимизации
- Будем использовать его как основной инструмент в рамках настоящего курса

Пример: раскраска карты Австралии в несколько цветов

Параметры и переменные в *MiniZinc*

```
% Это комментарий
% Целочисленный параметр – число цветов:
int: nc = 3;
% Для каждого штата зададим переменную,
% соответствующую его цвету:
var 1..nc: wa; var 1..nc: nt;
var 1..nc: sa; var 1..nc: q;
var 1..nc: nsw; var 1..nc: v;
var 1..nc: t;
```

Ограничения в *MiniZinc*

```
% Для каждой границы между странами задаем
% ограничение на неравенство цветов:
constraint wa != nt;
constraint wa != sa;
constraint nt != sa;
constraint nt != q;
constraint sa != q;
constraint sa != nsw;
constraint sa != v;
constraint q != nsw;
constraint nsw != v;
```

Решение задачи и вывод ответа

```
% решить задачу:
solve satisfy;
% вывести форматированный ответ:
output ["wa=", show(wa), "\t nt=", show(nt),
       "\t sa=", show(sa), "\n", "q=", show(q),
       "\t nsw=", show(nsw), "\t v=", show(v),
       "\n", "t=", show(t), "\n"];
```

```
Результат:
wa=1 nt=3 sa=2
q=1 nsw=3 v=1
t=1
```

Пример арифметической задачи

- Нужно приготовить несколько тортиков
- Тортики бывают банановые и шоколадные
- Запасы муки, бананов, сахара, масла и шоколада ограничены
- Различным типам тортиков нужны различные количества ингредиентов
- Хотим максимизировать прибыль от продажи

Параметры задачи

- Будем предполагать следующее:
- Для бананового тортика нужно 250 граммов муки, 2 банана, 75 граммов сахара и 100 граммов масла
- Для шоколадного тортика нужны 200 граммов муки, 75 граммов какао, 150 граммов сахара, 150 граммов масла.
- Стоимость бананового тортика – 400 рублей
- Стоимость шоколадного тортика – 450 рублей

Решение (1)

```
var 0..100: b; % число банановых тортиков
var 0..100: c; % число шоколадных тортиков

% ограничения на ингредиенты:
constraint 250*b + 200*c <= 4000; % мука
constraint 2*b <= 6; % бананы
constraint 75*b + 150*c <= 2000; % сахар
constraint 100*b + 150*c <= 500; % масло
constraint 75*c <= 500; % какао
```

Решение (2)

```
% Если стоимости тортиков – 400 и 450
% рублей соответственно, то нужно
% решить следующую задачу:
solve maximize 400*b + 450*c;
% Выведем ответ:
output ["no. of banana cakes = ", show(b),
     "\n", "no. of chocolate cakes = ",
     show(c), "\n"];
```

Множества и массивы

```
int: N;
set of int: mySet = 1..N;
array[1..N,1..N] of 0..N: arr1;
array[1..N,1..N] of 0..N: arr2;

% Более интересное ограничение с
% использованием агрегирующей функции:
constraint forall(i,j in mySet) (
   if arr1[i,j] > 0
   then arr1[i,j] = arr2[i,j]
   else true
   endif
);
```

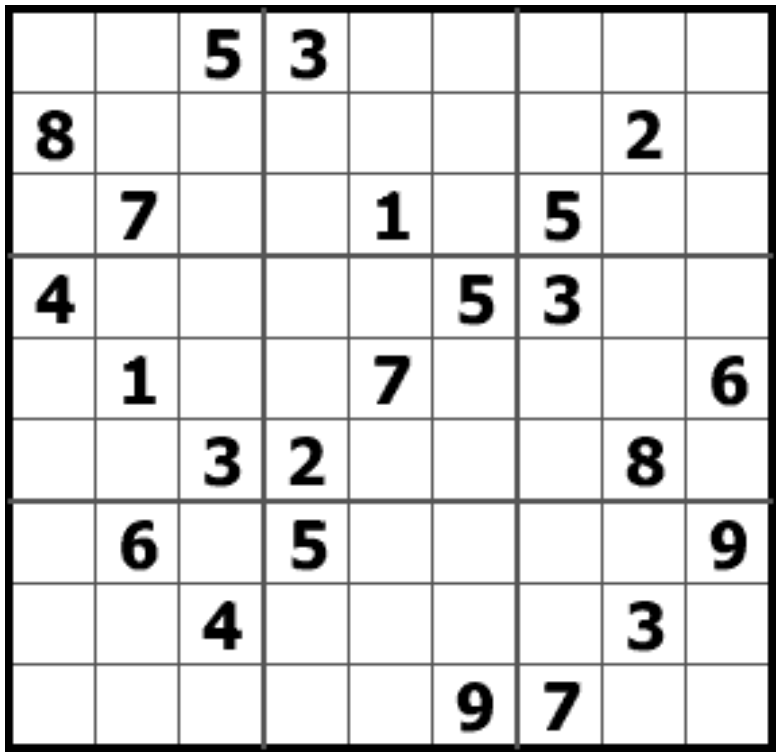
Агрегирующие функции

- forall – пример агрегирующей функции
- Еще есть:
  - min
  - max
  - sum
  - exists
  - xorall
  - iffall

Полноценный пример: судоку

Нужно расставить числа от 1 до 9 так, чтобы не было повторений:

- В стоках
- В столбцах
- В маленьких квадратах



Решение (1)

```
include "alldifferent.mzn";
int: S; % сторона маленького квадрата
int: N = S * S;
% число цифр для ответа:
int: digs = ceil(log(10.0,int2float(N)));
set of int: PuzzleRange = 1..N;
set of int: SubSquareRange = 1..S;
% поле (изначально пустое):
array[1..N,1..N] of 0..N: start;
% искомые переменные:
array[1..N,1..N] of var PuzzleRange: puzzle;
```

Решение (2)

```
% некоторые переменные уже известны:
constraint forall(i,j in PuzzleRange)(
      if start[i,j] > 0 then puzzle[i,j] =
      start[i,j] else true endif );
% числа в строках разные:
constraint forall (i in PuzzleRange) (
      alldifferent( [ puzzle[i,j] | j in PuzzleRange
      ]) );
% числа в столбцах разные:
constraint forall (j in PuzzleRange) (
      alldifferent( [ puzzle[i,j] | i in PuzzleRange
      ]) );
```

Решение (3)

```
% числа в маленьких квадратах разные:
constraint forall (a, o in SubSquareRange)(
      alldifferent( [ puzzle[(a-1) *S + a1, (o-1)*S
      + o1] | a1, o1 in SubSquareRange ] ) );

solve satisfy;

output [ show_int(digs,puzzle[i,j]) ++ " " ++
      if j mod S == 0 then " " else "" endif ++
      if j == N /\ i != N then
      if i mod S == 0 then "\n\n" else "\n" endif
      else "" endif | i,j in PuzzleRange ] ++
      ["\n"];
```

Задание начального поля

```
S=3;
start=[|
0, 0, 0, 0, 0, 0, 0, 0, 0|
0, 6, 8, 4, 0, 1, 0, 7, 0|
0, 0, 0, 0, 8, 5, 0, 3, 0|
0, 2, 6, 8, 0, 9, 0, 4, 0|
0, 0, 7, 0, 0, 0, 9, 0, 0|
0, 5, 0, 1, 0, 6, 3, 2, 0|
0, 4, 0, 6, 1, 0, 0, 0, 0|
0, 3, 0, 2, 0, 7, 6, 9, 0|
0, 0, 0, 0, 0, 0, 0, 0, 0|];
```

MiniZinc Challenge 2013

- http://www.minizinc.org/challenge2013/challenge.html
- Соревнование солверов на 100 задачах в формате *MiniZinc*
- Медалисты: *Opturion/CPX*, *OR-Tools*, *Choco*, *gecode*, *izplus*

Заключение

- *MiniZinc* – один из языков для формулировки задач удовлетворения ограничений
- Подмножество более высокоуровнего языка *Zinc*
- Реализации MiniZinc: *Gecode/FlatZinc*, *ECLiPSe*, *SICStus Prolog*, *JaCoP*, *SCIP*, *Opturion CPX*, *MinisatID*

Спасибо за внимание!
Вопросы?