Implementing Tuple Variables in Gecode

Bachelor Thesis

Patrik Broman

Constraint Programming

- Declarative style programming
- The programmer states what must hold, without stating how it is achieved
- A constraint programming variable has more in common with mathematical variables than regular variables used in programming

Constraint Programming

```
while not done:
    propagators prune as much as possible
    a brancher assignes a value to a variable
```

Constraint Programming

Sudoku is a good example of a problem suitable for constraint programming.

Constraint Programming

```
IntVar s[9][9] = {1, … ,9}

for x=1 to 9:
   distinct(s[x][1], s[x][2], … , s[x][9])

for y=1 to 9:
   distinct(s[1][y], s[2][y], … , s[9][y])

for x=1 to 3:
   for y=1 to 3:
      distinct( s[3x-2][3y-2], s[3x-1][3y-2], s[3x][3y-2],
                s[3x-2][3y-1], s[3x-1][3y-1], s[3x][3y-1],
                s[3x-2][3y], s[3x-1][3y], s[3x][3y])
```

What data types exist in Gecode?

- Boolean
- Integer
- Integer set
- Float

What about Tuples?

What about Tuples?

Consider the following example:

```
Intvar a,b = {1, …, 1000}
rel(a ≠ b)
```

What about Tuples?

- Without tuples, it is impossible to prune a combination of values.
- Tuples move work from branchers to propagators.

Pair Variables

- We restrict tuples to 2-tuples and call them pairs
- Two variants are created
  - Exact domain representation
  - Approximate representation

Exact Pair

- Stores a long list of ALL combinations
- Requires more memory
- Some operations are slower
- Can be pruned more efficiently

Approximate Pair

- Does not store all possible combinations
- Requires less memory
- Some operations are faster
- Can not be pruned as efficiently as the exact representation

Comparison