Introduction to Gecode

- Notes based on manual at: http://www.gecode.org/doc-latest/MPG.pdf
  - Ch. 1-2.3.1
- Gecode features
- Modeling and introductory example
- Compiling (Visual Studio)

Gecode goals

- Support constraint-based systems, applications
  - Need to support users who are not primarily programmers
  - Need to support modeling of problems
- Follows C++ standard to maximize portability
  - Does require Visual Studio on Windows

Terms

- *Space* = contains *variables*, *propagators*, *branchers*, *order*
- Propagator = implementation of constraint
- Brancher = *labeling*, describes search tree’s shape
- Order = way to compare solutions

Defining a Model

- Define a subclass of Space class. Implement
  - Constructor: implements model
  - Copy constructor, copy function: required for search

Introductory example

- Cryptarithmetic problem: SEND + MORE = MONEY
- Implementation:

```
#include <gecode/int.hh>
```

  - For integer constraints

```
#include <gecode/search.hh>
```

  - To access search engine to find solution

```
using namespace Gecode;
```

  - Identifiers are contained here

Introductory example

- Subclass of Space:

```
class SendMoreMoney : public Space {
```

- Create array for variables. Values are integers, so use `IntVarArray`

```
protected:
  IntVarArray l;
```

Constructor

- Constructor
  - Constructing `IntVarArray` passes `Space` containing array, number of variables, min and max values
  - Set up convenient aliases for the variables

```
public:
  SendMoreMoney(void) : l(*this, 8, 0, 9) {
    IntVar s(l[0]), e(l[1]), n(l[2]), d(l[3]),
           m(l[4]), o(l[5]), r(l[6]), y(l[7]);
```

Relation, distinct constraints

- Constraints to prevent leading zeros
  - Gecode calls functions like `rel()` *constraint post functions*
    - `rel()` is a set of functions for relation constraints
    - `IRT_NQ` indicates variables are “not equal” (to last arg, 0)
  - (syntax is somewhat declarative)

```
rel(*this, s, IRT_NQ, 0);
rel(*this, m, IRT_NQ, 0);
```

  - `distinct()` requires all values in 2nd arg to be distinct

```
distinct(*this, l);
```

Linear constraint

- Create sum to represent SEND + MORE - MONEY
  - `IntArgs` represents coefficient array
  - `IntVarArgs` represents variable array

```
IntArgs c(4+4+5); IntVarArgs x(4+4+5);
  c[0]=1000; c[1]=100; c[2]=10; c[3]=1;
  x[0]=s;    x[1]=e;   x[2]=n;  x[3]=d;
  c[4]=1000; c[5]=100; c[6]=10; c[7]=1;
  x[4]=m;    x[5]=o;   x[6]=r;  x[7]=e;
  c[8]=-10000; c[9]=-1000; c[10]=-100; c[11]=-10;
c[12]=-1;
  x[8]=m;      x[9]=o;     x[10]=n;    x[11]=e;
x[12]=y;
```

Linear constraint

- Now create constraint
  - `IRT_EQ` to indicate sum c[0]*x[0] + c[1]*x[1] + c[2]*x[2] + … = 0

```
linear(*this, c, x, IRT_EQ, 0);
```

Branching

- Indicate variables to assign, strategy for picking variables, strategy for picking values
  - `INT_VAR_SIZE_MIN()` picks the variable with the smallest domain
  - `INT_VAL_MIN()` picks the smallest value
  - Can have multiple branchers, which are used in the order they are “posted”

```
branch(*this, l, INT_VAR_SIZE_MIN(),
INT_VAL_MIN());
```

Cloning

- Gecode search uses *recomputation* and *cloning*
  - Need to support cloning of Space's (like Java's clone() method)
  - Implement a copy constructor:
    - Call copy constructor of parent class
    - `share` parameter is true if copies must be used within the same thread because they share data, false if can be used in different threads

```
  SendMoreMoney(bool share, SendMoreMoney& s) :
Space(share, s) {
    l.update(*this, share, s.l);
  }
```

Cloning (copy function)

- Implement a function to support copying:

```
virtual Space * copy(bool share) ) {
    return new SendMoreMoney(share,*this);
  }
```

  - Virtual to allow search engine code to not have to know which subclass is returned

Printing solution

- Gecode overrides << to support displaying solutions to console
  - In general, other code would access solution variable to process it

```
void print(void) const {
   std::cout << l << std::endl;
}
```

main()

- Simple code to search for all solutions and display them:

```
int main(int argc, char* argv[]) {
  // create model and search engine
  SendMoreMoney* m = new SendMoreMoney;
  DFS<SendMoreMoney> e(m);
  // e will use clone of m to allow other search
  // engines to use m, but only 1 engine here
  delete m;
```

main()

```
// search and print all solutions
  while (SendMoreMoney* s = e.next()) {
    s->print(); delete s;
  }
  return 0;
}
```

Compiling

- Visual Studio (2013/2015/2017) command line
  - Windows Start button > Visual Studio 2017 > Developer Command Prompt
  - If running 32-bit Visual Studio (ex: many community editions),
    - Use cross-compiling feature to generate code for 64-bit architecture
    - At command prompt, `vcvarsall amd64`
      - `vcvarsall` is under “VC” subfolder of Visual studio (if present, `vcvars64` will also work)
      - Ex: for VS 2017, C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Auxiliary\Build
    - Gecode binary distributions target 64-bit architecture
  - Change folder to where file is located

Compiling

- To compile example source:
  - /Ox – middle char is uppercase letter O

```
cl /DNDEBUG /EHsc /MD /Ox /wd4355
-I"%GECODEDIR%\include" -c -Fosend-more-money.obj
-Tpsend-more-money.cpp
```

- To link object file

```
cl /DNDEBUG /EHsc /MD /Ox /wd4355
-I"%GECODEDIR%\include" -Fesend-more-money.exe send-
more-money.obj /link /LIBPATH:"%GECODEDIR%\lib"
```

Compiling

- If not using VS for other classes, consider setting CL environment variable to

`/DNDEBUG /EHsc /MD /Ox /wd4355 -I"%GECODEDIR%\include“`

- Reduces typing when you open new VS developer command window

Results

- {9, 5, 6, 7, 1, 0, 8, 2}
  - S = 9
  - E = 5
  - N = 6
  - D = 7
  - M = 1
  - O = 0
  - R = 8
  - Y = 2
  - 9567 + 1085 = 10652

Some things to try

- Experiment with propagation:
  - Instead of creating DFS:
    - Display result of initial constraints

```
m->print();
```

    - Output shows domains of the variables, minus 0's for m and s variables

```
{[1..9], [0..9], [0..9], [0..9], [1..9], [0..9],
[0..9], [0..9]}
```

Some things to try

- Propagate constraints one time

```
m->status();
```

- Display updated values

```
m->print();
```

- Assigns values to s, m and o variables, removes values from other variables:

```
{9, [4..7], [5..8], [2..8], 1, 0, [2..8],
[2..8]}
```

Handling exceptions

- Examples skip this for readability, but in general, should catch:

```
try { … }
catch(Exception e) {
  std::cerr << "Gecode exception: " <<
    e.what() << std::endl;
}
return 0;
```