Gecode

an open constraint solving library

Christian Schulte
KTH Royal Institute of Technology, Sweden

Gecode People

- Core team
  - Christian Schulte, Guido Tack, Mikael Z. Lagerkvist.
- Code
  - contributions: Christopher Mears, David Rijsman, Denys Duchier, Filip Konvicka, Gabor Szokoli, Gabriel Hjort Blindell, Gregory Crosswhite, Håkan Kjellerstrand, Joseph Scott, Lubomir Moric, Patrick Pekczynski, Raphael Reischuk, Stefano Gualandi, Tias Guns, Vincent Barichard.
  - fixes: Alexander Samoilov, David Rijsman, Geoffrey Chu, Grégoire Dooms, Gustavo Gutierrez, Olof Sivertsson, Zandra Norman.
- Documentation
  - contributions: Christopher Mears.
  - fixes: Seyed Hosein Attarzadeh Niaki, Vincent Barichard, Pavel Bochman, Felix Brandt, Markus Böhm, Roberto Castañeda Lozano, Gregory Crosswhite, Pierre Flener, Gustavo Gutierrez, Gabriel Hjort Blindell, Sverker Janson, Andreas Karlsson, Håkan Kjellerstrand, Chris Mears, Benjamin Negrevergne, Flutra Osmani, Max Ostrowski, David Rijsman, Dan Scott, Kish Shen.

Gecode

Generic Constraint Development Environment

- **open**
  - easy interfacing to other systems
  - supports programming of: constraints, branching strategies, search engines, variable domains
- **comprehensive**
  - constraints over integers, Booleans, sets, and floats
    - different propagation strength, half and full reification, …
  - advanced branching heuristics (accumulated failure count, activity)
  - many search engines (parallel, interactive graphical, restarts)
  - automatic symmetry breaking (LDSB)
  - no-goods from restarts
  - MiniZinc support

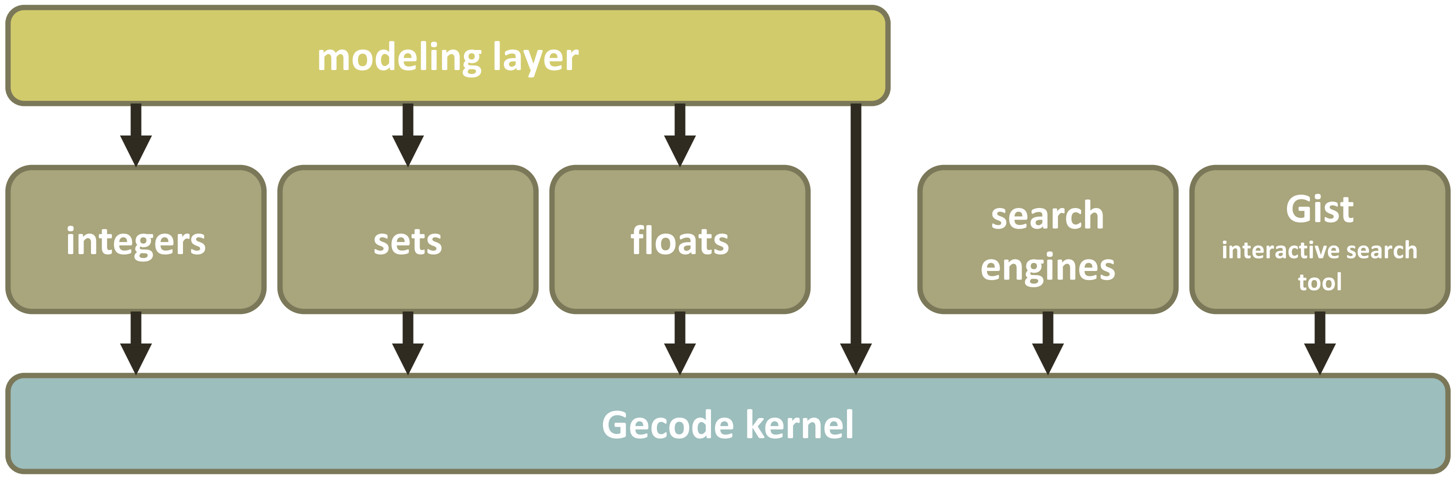
Gecode

Generic Constraint Development Environment

- **efficient**
    - *all* gold medals in *all* categories at *all* MiniZinc Challenges
- **documented**
    - tutorial (> 500 pages) and reference documentation
- **free**
    - MIT license, listed as free software by FSF
- **portable**
    - implemented in C++ that carefully follows the C++ standard
- **parallel**
    - exploits multiple cores of today's hardware for search
- **tested**
    - some 50000 test cases, coverage close to 100%

SOME BASIC FACTS

Architecture



- Small domain-independent kernel
- Modules
  - per variable type: variables, constraint, branchings, …
  - search, FlatZinc support, …
- Modeling layer
  - arithmetic, set, Boolean operators; regular expressions; matrices, …
- All APIs are user-level and documented (tutorial + reference)

Openness

- MIT license permits commercial, closed-source use
    - motivation: public funding, focus on research
    - not a reason: attitude, politics, dogmatism
- More than a license
    - **license** restricts what users **may do**
    - **code and documentation** restrict what users **can do**
- Modular, structured, documented, readable
    - complete tutorial and reference documentation
    - new ideas from Gecode available as scientific publications
- Equal rights: Gecode users are first-class citizens
    - you can do what we can do: APIs
    - you can know what we know: documentation
    - on every level of abstraction

Constraints in Gecode

- Constraint families
  - arithmetics, Boolean, ordering, ….
  - alldifferent, count (global cardinality, …), element, scheduling, table and regular, sorted, sequence, circuit, channel, bin-packing, lex, geometrical packing, nvalue, lex, value precedence, …
- Families
  - many different variants and different propagation strength
- All global constraints from MiniZinc have a native implementation
- Gecode ⇆ Global Constraint Catalogue: > 70 constraints

  abs_value, all_equal, alldifferent, alldifferent_cst, among, among_seq, among_var, and, arith, atleast, atmost, bin_packing, bin_packing_capa, circuit, clause_and, clause_or, count, counts, cumulative, cumulatives, decreasing, diffn, disjunctive, domain, domain_constraint, elem, element, element_matrix, eq, eq_set, equivalent, exactly, geq, global_cardinality, gt, imply, in, in_interval, in_intervals, in_relation, in_set, increasing, int_value_precede, int_value_precede_chain, inverse, inverse_offset, leq, lex, lex_greater, lex_greatereq, lex_less, lex_lesseq, link_set_to_booleans, lt, maximum, minimum, nand, neq, nor, not_all_equal, not_in, nvalue, nvalues, or, roots, scalar_product, set_value_precede, sort, sort_permutation, strictly_decreasing, strictly_increasing, sum_ctr, sum_set, xor

History

- 2002
  - development started
- 1.0.0
  - December 2005
- 2.0.0
  - November 2007
- 3.0.0
  - March 2009
- 4.0.0
  - March 2013
- 4.2.0 (current)
  - July 2013

43 kloc, 21 klod

77 kloc, 41 klod

34 releases

81 kloc, 41 klod

164 kloc, 69 klod

168 kloc, 71 klod

Tutorial Documentation

- 2002
  - development started
- 1.0.0 — 43 kloc, 21 klod
  - December 2005
- 2.0.0 — 77 kloc, 41 klod
  - November 2007
- 3.0.0 — 1 klod
  - March 2009
- 4.0.0 — 164 kloc, 69 klod
  - March 2013
- 4.2.0 (current)
  - July 2013

Modeling with Gecode (98 pages)

Modeling & Programming with Gecode (522 pages)

Future

- Large neighborhood search and other meta-heuristics
    - contribution expected
- Simple temporal networks for scheduling
    - contribution expected
- More expressive modeling layer on top of libmzn
- Grammar constraints
    - contribution expected
- Propagator groups
- …

- Contributions anyone?

Initial Goals

- Research
  - architecture of constraint programming systems
  - propagation algorithms, search, modeling languages, ...
- Efficiency
  - competitive
  - proving architecture right
- Education
  - state-of-the-art, free platform for teaching
- CP community service
  - provide an open platform for research (think back to 2002!)

Users

- Research
  - experiments, comparison, extensions, …
  - Google scholar: 924 references to Gecode [Aug 2013]
- Education
  - KTH, Uppsala U, U Freiburg, UC Louvain, Saarland U, American U Cairo, U Waterloo, U Javeriana-Cali, U Udine, U Vienna, …
- Industry and other systems
  - several companies have integrated Gecode into products (part of hybrid solvers)
  - Gecode as starting point for other systems
  - parts of Gecode's code incorporated into other systems

Use Cases: Research

- Benchmarking platform for models: lots of people…
  - CP 2013:
    1. Leo, Mears, ea, Globalizing Constraint Models
    2. Gaspero, Rendl, ea: Constraint-Based Approaches for Balancing Bike Sharing Systems
    3. Lagerkvist, Nordkvist, ea: Laser Cutting Path Planning Using CP
    4. Mann, Nahar, ea: Atom Mapping with Constraint Programming

- Benchmarking platform for implementations: lots of people…
  - requires open source (improve what Gecode implements itself)
  - MiniZinc challenge 2013: Gecoxicals (replace constraints with generated ones)
  - CP 2013:
    5. Gualandi, Lombardi: A Simple and Effective Decomposition for the Multidimensional Binpacking Constraint
    6. He, Flener, ea: Solving String Constraints: The Case for Constraint Programming
    7. Loth, Sebag, ea: Bandit-Based Search for Constraint Programming
    8. Régin, Rezgui, ea, Embarrassingly Parallel Search

- Gecode example models as reference
  - Castineiras, De Cauwer, O'Sullivan, Weibull-based Benchmarks for *Bin Packing*. CP 2012.

- Base system for extensions
  - Qecode: quantified constraints (Benedetti, Lalouet, Vautard)
  - Gelato: hybrid of propagation and local search (Cipriano, Di Gaspero, Dovier)
  - Gecode interfaces powerful enough: no extension required

Use Cases: Education

- Courses feasible that include
  - modeling
  - principles

  but also
  - programming search heuristics (branchers)
  - programming constraints (propagators)

- Essential for programming
  - accessible documentation…
  - …including many examples

Use Cases: Interfacing

- Quintiq integrates Gecode as CP component
    - available in their modeling language
- Cologne: A Declarative Distributed Constraint Optimization Platform
    - U Penn, AT&T Labs, Raytheon
    - Datalog + constraints in distributed setup [Liu ea, VLDB 2012]
- Constraint Programming for Itemset Mining (CP4IM)
    - declarative approach to constraint-based itemset mining [Guns, Nijssen, De Raedt, KU Leuven]
- Whatever language
    - modeling: AMPL, MiniZinc, …
    - programming: Java, Prolog (> 1), Lisp (> 1), Ruby, Python (> 1), Haskell, …

Use Cases: Other Systems

- Parts of Gecode integrated into other systems
    - Caspar (global constraint implementations)
    - Google or-tools
    - possibly more: that’s okay due to MIT license
- Gecode as starting point for other systems
    - Opturion’s CPX Discrete Optimizer
    - definitely more: that’s okay due to MIT license

Deployment & Distribution

- Open source ≠ Linux only
    - Gecode is native citizen of: Linux, Mac, Windows
- High-quality
    - extensive test infrastructure (around 16% of code base)
- Downloads from Gecode webpage
    - software: between 25 to 125 per day (total > 40000)
    - documentation: between 50 to 300 per day
- Included in
    - Debian, Ubuntu, Fedora, OpenSUSE, Gentoo, FreeBSD, …

Integration & Standardization

- Why C^{++} as implementation language?
  - good compromise between portability and efficiency
  - good for interfacing

well demonstrated

- Integration with XYZ…
  - Gecode empowers users to do it
  - no "Jack of all trades, master of none"

well demonstrated

- Standardization
  - any user can build an interface to whatever standard…
  - systems are the wrong level of abstraction for standardization
  - MiniZinc and AMPL are de-facto standards

Golomb ruler (CSPlib problem 006) à la Gecode

MODELING

Golomb Ruler: Model

```
Golomb(int n) : m(*this, n, 0, Int::Limits::max) {









}
```

- Declare n variables with values between 0 and largest integer value

Golomb Ruler: Model

```
Golomb(int n) : m(*this, n, 0, Int::Limits::max) {
  rel(*this, m[0] == 0);
  rel(*this, m, IRT_LE);













}
```

- Constrain first mark `m[0]` to `0`
- Constrain marks `m` to be strictly increasing (`IRT_LE` = integer relation type for less)

Golomb Ruler: Model

```
Golomb(int n) : m(*this, n, 0, Int::Limits::max) {
  rel(*this, m[0] == 0);
  rel(*this, m, IRT_LE);

  IntVarArgs d;
  for (int k=0, i=0; i<n-1; i++)
    for (int j=i+1; j<n; j++, k++)
      d << expr(*this, m[j] - m[i]);
  distinct(*this, d);


}
```

- Collect variables for distances between marks in d
- Constrain d to be all different (`distinct` in Gecode)

Golomb Ruler: Model

```
Golomb(int n) : m(*this, n, 0, Int::Limits::max) {
  rel(*this, m[0] == 0);
  rel(*this, m, IRT_LE);

  IntVarArgs d;
  for (int k=0, i=0; i<n-1; i++)
    for (int j=i+1; j<n; j++, k++)
      d << expr(*this, m[j] - m[i]);
  distinct(*this, d);

  branch(*this, m, INT_VAR_NONE(), INT_VAL_MIN());
}
```

- Branch on marks `m` with no selection strategy (left-to-right) and try the smallest value first

Golomb Ruler: Cost Function

```
IntVar cost() const {
  return m[m.size()-1];
}
```

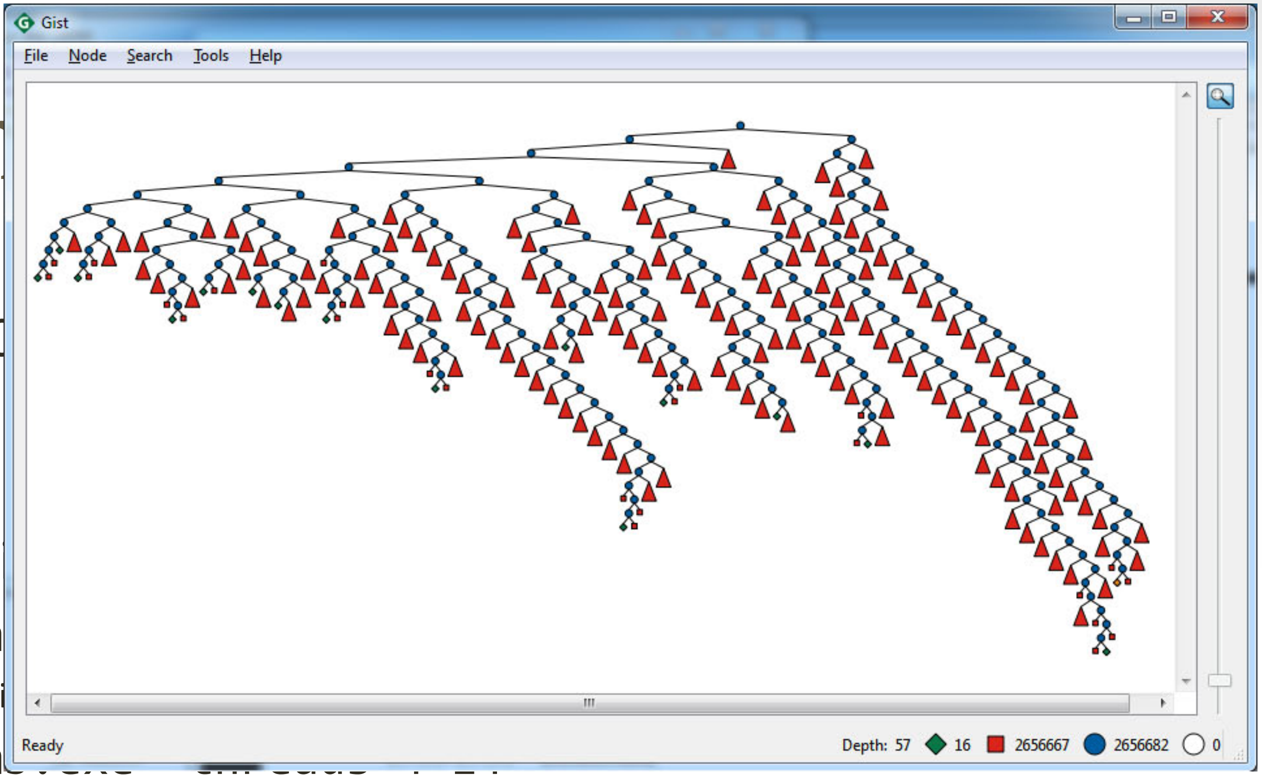
- Return last mark as cost

Runnin

- Some 10-2C
  - followi
- After comp
  - golom
    - fi
  - golom
    - use four threads for parallel search for 12 marks
  - `golomb.exe –mode gist 12`
    - use graphical interactive search tool (scales to millions of nodes)

or

- use restarts… `-restart luby`
- use no-goods from restarts… `-no-goods 1`
- lots more (too many as some people say… ☹)



```
Ready                    Depth: 57  ◆ 16  ■ 2656667  ● 2656682  ○ 0
```

Summary

- Gecode is…

open	comprehensive	efficient
documented	free	portable
parallel	tested	

…and pretty cool for…

modeling	solving
programming	interfacing