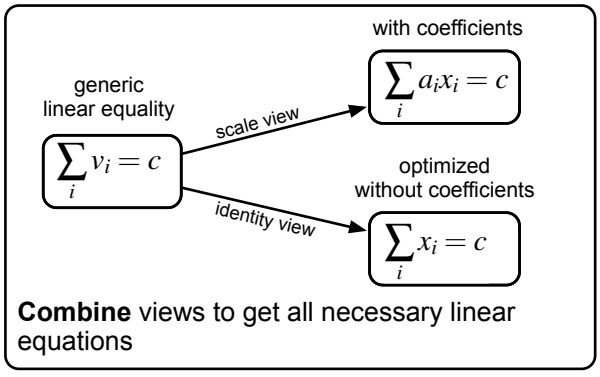
Views and Iterators for Generic Constraint Implementations

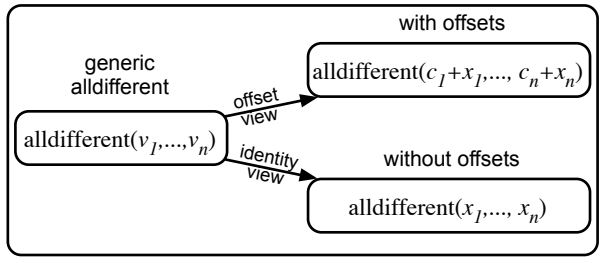
Christian Schulte, KTH, Sweden
Guido Tack, Saarland University, Germany

Generic Propagators

Instantiate one propagator implementation with different variable views:



Combine views to get all necessary linear equations



- reuse of propagation algorithms
- separation of concerns (e.g. linear equations vs. rounding)

Variable Views

A view is the fundamental **datatype** propagators operate on.
All views have a **uniform interface**.

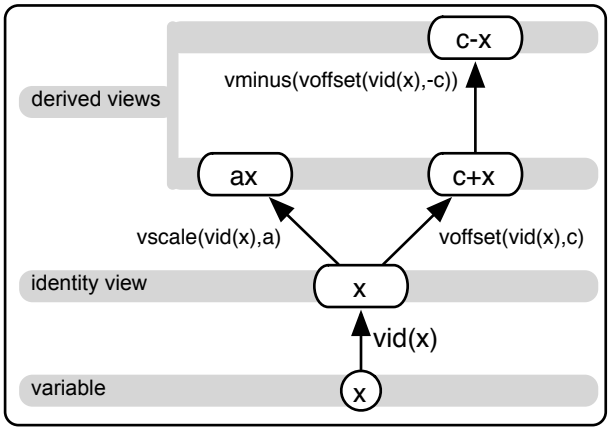
Views for **integer variables**:
- scale with a constant
- add constant offset
- negate

Views for **set variables**:
- complement

Inter-domain views:
- integer viewed as singleton set

Views **without variables**:
- integer and set constants

Views are generic: combine e.g. negation and offset.



Range Iterators

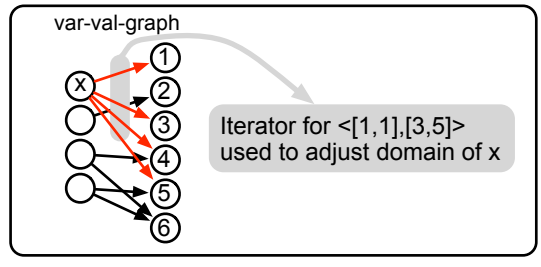
Represent sets of integers as range sequences. Range iterators access ranges of a sequence one after the other.

Domain operations
Domain of a variable updated with range operations, taking an iterator as argument. Simultaneous removal of several values.

Iterator combinators
Compute union, intersection etc. of iterators. Result is again an iterator. No memory needed for intermediate results.

Iterators for finite set constraints
provide all basic operations on sets. Efficient propagators, no intermediate results.

Iterators as adaptors:



Implementation

The presented architecture is an **orthogonal layer of abstraction** for any CP system.

Based on *parametric polymorphism:*
C++ templates, Java generics, ML functors

Using C++ templates:
monomorphization allows for aggressive optimization. Views incur no run-time overhead.

all presented concepts are implemented in Gecode

www.gecode.org