Constraints @ KTH

Sven Lämmermann
Christian Schulte
Thomas Sjöland

IMIT, KTH – Royal Institute of Technology
Stockholm, Sweden
laemmi,schulte,sjoland@imit.kth.se

Main Focus @ KTH

- Programming languages
  - design
  - implementation
  - program analysis
  - program synthesis

People @ KTH

Seif Haridi	seif@sics.se	
Sven Lämmermann	laemmi@imit.kth.se	
Christian Schulte	schulte@imit.kth.se	
Thomas Sjöland	sjoland@imit.kth.se	here!

Projects and Interests @ KTH

- Oz and Mozart
- Program analysis
  - analysis of deep-guard ccp programs
  - analysis of `clp(FD)` programs
- Program synthesis
  - logic-based program synthesis
  - synthesis of concurrent programs
- Constraint libraries: Gecode

Oz and Mozart

- Some facts
- Concurrency and distribution
- Problem solving

Oz and Mozart

- Constraint-based programming system
  - concurrent and distributed programming
  - combinatorial problem solving
  - and combinations: intelligent agents, ...
- People

  Seif Haridi seif@sics.se
  Christian Schulte schulte@imit.kth.se
  Thomas Sjöland sjoland@imit.kth.se

  plus many other people @ KTH/SICS

Mozart Fact Sheet

- Mozart implements Oz
  - concurrent constraint programming language
  - with: objects, functions, threads, programmable search
- Developed by Mozart Consortium
  - Saarland University, Germany
  - SICS/KTH, Sweden
  - Université catholique de Louvain, Belgium

Mozart Fact Sheet

- Freely available at
  - www.mozart-oz.org
- Many platforms supported
  - Unix Windows Mac OS
- Active user community
- Comes with extensive documentation
- Many applications

Constraints for Concurrency

- Constraints
  - describe data structures
  - used for control [as opposed to Prolog]
- Logic variables as dataflow variables
  - unconstrained ⇒ suspension
  - constrained ⇒ resumption
  - synchronization is automatic
- Well established idea
  - resumption condition is logical entailment
    - [Maher,87], ccp [Saraswat,90]

Constraints for Distribution

- Distribution presupposes concurrency
  - hey, at least two threads
- Logic variables
  - serve as communication channel
  - provide latency tolerance: synchronization
    - is automatic
    - only if necessary
  - support incremental message
    - composition
    - reception and processing
  - make distribution orthogonal aspect
- Obtained by distribution protocols
  - [Haridi et al., 99]

Research in Distribution

- Distribution protocols for other entities
  - in particular objects
- Fault tolerance
- Distributed simulation
- Peer-to-peer computing [EU Pepito]
- Bring protocols to distribution library
- Check: www.sics.se/dsl

Problem Solving

- Constraint domains
  - tree constraints (records, feature)
  - finite domains
  - finite sets
- Programmable search and combinators
  - based on computation spaces
  - makes search compatible with concurrency
  - new book:
    - Christian Schulte, Programming Constraint Services
    - LNAI 2302, Springer-Verlag, 2002

Combination

- Excellent choice for multi-agent systems!
- Application to parallel search
  - programmable search
  - parallelism through distribution
    [Networked Personal Computers]

Program Analysis

- Hierarchical CCP
- Optimizing `clp(FD)` programs

Analysis of Hierarchical CCP

- Analysing ccp programs
  - nested spaces (deep guards)
  - domain independent analysis framework
- People
  Thomas Sjöland `sjoland@imit.kth.se`

Why Analysis of Hierarchical CCP?

- Concurrent programs matter
  - important for distribution
  - encapsulation by spaces mandatory
- Provide **global** information
  - compiler ⇒ efficiency
  - program transformation ⇒ efficiency
  - programmer ⇒ understanding

How To Do Analysis?

- Formulate abstract semantics
- Relate it to concrete semantics
  - safeness
  - termination
  - accuracy
- Devise framework approach
  - generic for abstract constraint domains
  - in particular: abstract entailment

Optimizing `clp(FD)` Programs

- Optimize `clp(FD)` programs
  - using cheaper bounds propagation
  - retaining same search space
- People
  Christian Schulte `schulte@imit.kth.se`
  Peter J. Stuckey `pjs@cs.mu.oz.au`
  [University of Melbourne, Australia]

Use Bound Propagation!

- Domain propagation
  - good pruning ⇒ high cost
  - less pruning ⇒ efficient
- Use bound propagation
  - when provable that search space the same

Approach

- Establish notions of equivalence
  - set of constraints can be replaced
  - common finite domain constraints considered
- Analyse and optimize programs
  - abstract interpretation of `clp(FD)` programs
  - replace domain by bounds propagation if equivalent

Status

- Paper: [PPDP,2001]
- First experiments show
  - encouraging applicability
  - good speedup
- Current work
  - apply at runtime
  - better applicability

Program Synthesis

- Logic-based program synthesis
- Synthesis of concurrent programs

Logic-based Prog. Synthesis and Constraint Networks

- Constraint networks have various logical interpretations
  - FCNs have a good explanation in terms of constructive logic
  - Value propagation (VP) algorithm works also on higher-order functional constraint networks
    - Has relation to intuitionistic propositional logic (IPL)
    - Steps of value propagation can be regarded as valid proof rules of IPL

Synthesis of Concurrent Programs

- VP can be pre-planned on constraint network schemas (CNS)
  - Abstracting away from concrete interpretations of constraints at the planning stage
  - Interesting for deriving concurrent implementations of synthesized programs
  - Done for IPL
  - Can VP on CNSs be a useful technique for program synthesis using (possibly a subset of) inuitionistic first-order logic?

Constraint Libraries

Gecode

- Generic environment for developing constraints
- People:

Christian Schulte schulte@imit.kth.se
Martin Henz henz@comp.nus.edu.sg
[National University of Singapore]

What Is Gecode?

- Environment to develop constraint programming libraries
  - domain independent
- Collection of search engines
- Implementation of finite domain constraint library
- ...

Gecode Motivation

- Systems geared mainly at deployment
- Lack of
  - published models for systems
  - published implementation techniques
  - support for extension
- Transfer ideas from Mozart
  - to language neutral setting
  - search and combinators

Gecode Goals: Model

- Understanding constraint propagation
- Devise simple abstract model
  - includes optimizations
  - prove model correct
- Starting point: [Apt, Toplas 2001]

Gecode Goals: Implementation

- Focus on constraint propagation
- Devise general techniques
  - parametric propagation algorithms
  - incremental constraint propagation
- High-performance
  - simple and expressive architecture
- Distinguish domain-independent and domain-dependent aspects

Gecode Goals: Usage

- Use the finite domain library
  - add new constraints
- Do a new constraint library
  - add new variable domains
  - add new constraints
- Hope is to reuse
  - existing abstractions
  - existing algorithms

Gecode Non-Goals

- Coding level optimizations
- Coverage of standard constraints
- Modeling support
  - known to be orthogonal
  - for example:
    - OPL [Van Hentenryck, 1999]
    - Modeler++ [Michel&Van Hentenryck, 2001]

Gecode Fact Sheet

- C++ based
  - heavy use of polymorphism (templates)
- Available upon request
- No modeling support
- Experimental in nature
- Copying-based search
  - adaptive + batch recomputation

Gecode Status

- Features finite domain constraints
  - modest set of constraints
- First steps in formal model
- New general implementation techniques
- Encouraging performance
  - 2 to 10-times faster than Mozart

Interested…

- Get in touch…
  - … here
  - … by help of SweConsNet