CP Solvers/Learning Constraint Programming

CP 2013, 2013-09-16, Uppsala, Sweden
Workshop
“CP Solvers: Modeling, Applications, Integration, and Standardization”

Håkan Kjellerstrand (hakank@gmail.com)
Independent Researcher, Malmö, Sweden

http://www.hakank.org/

My Constraint Programming Blog:
http://www.hakank.org/constraint_programming_blog/

The talk

- Some introduction about me and what I do

- Some thoughts about CP and the role of CP developers
  for making CP more known outside of the research field

- Learning CP: A subjective comparison matrix of the tested solvers.
  Will not go into too much details...

Some more about me

- Have been actively interested in/fascinated by CP since 2008

- CP is a private interest
  That might explain some of my preferences and ideas...

- Not a theory guy, much more a modeling guy.
  “Grey box modeler”

- Often quite small problems
  mathematical puzzles, discrete mathematics, grid puzzles,
  combinatorial problems, planning puzzles, etc.

- Still looking for the “perfect tool”

What do I do?

Testing a CP system generally follows this scheme:

- learning the host language if needed

- systematically writing “learning problems” (from ~20 .. ~200)

- focus on certain (syntactic) issues/features

- contacting the developers: bug report/complaining

- publishing: “My System X page”, “A first look a System X”

Tested ~25 CP Systems

- Choco (21 models)
- ECLiPSe CLP (177 models)
- Gecode/R (29 models)
- JaCoP/Scala (39 models)
- SICStus Prolog (152 models)
- Essence'/Savile Row (56 models)
- Google or-tools/Python (202 models)
- Google or-tools/C# (129 models)
- Java JSR-331 (42 models)
- AIMMS+CP (39 models)
- Choco3 (90 models)
- Picat (~250 CP models)
- ILOG CP Optimizer OPL (~110 models, not yet published)
- Answer Set Programming ("related paradigm", 87 encodings)
- Comet (168 models)
- Gecode (165 models)
- JaCoP (18 models)
- MiniZinc (1046 models)
- Essence'/Tailor (26 models)
- Zinc (39 models)
- Google or-tools/Java (36 models)
- OscaR (Scala in OR) (146 models)
- Numberjack (52 models)
- B-Prolog (207 models)
- AMPL (~100 "pure" CP models)

And ~22 FlatZinc solvers.

Solvers at the CP Solver Workshop

Almost all the solvers at this workshop I've tested (some more than other):

- Gecode, Choco3, CP Optimizer/OPL, JaCoP, OR-tools,
  JSR-331, AMPL+CP, IZ-C (as FlatZinc solver), Numberjack,
  AIMMS+CP
  See http://www.hakank.org/constraint_programming/

Not tested:

- Xpress-Kalis
  However: in 2007 I ported most of Martin Chlond's IP models written
  in Xpress Mosel (and Kalis) to AMPL.
  So I've been – in part - exposed to it.

CP Solver Catalog

CP Solver Catalog, created by Jacob Feldman for this Workshop

http://openjvm.jvmhost.net/CPSolvers/

'''

This catalogue contains detailed profiles of constraint programming tools submitted by their developers to this website.
The submitters have an exclusive access and are solely responsible for the quality of the provided information about their tools.

'''

Common Constraint Programming Problems

http://hakank.org/common_cp_models/

* I always start testing a CP system with a number of standard "learning problems" to get a feeling for different constructs in the CP system.

  [Implementing `alldifferent_except_0` is often a good proxy.]

* 24 systems officially published (at least one system is still unpublished):
  Currently 430 problems are implemented in at least 2 CP systems.
  There are 2424 implemented models.
  Total implemented models, including those not showed at the page, is about 3633.

  There is about 159 problems implemented in >= 6 systems.

* All published models are collected at my GitHub repo:
  https://github.com/hakank/hakank/

Learning Constraint Programming: Why?

Why learning Constraint Programming?

- CP it is an excellent tool for certain type of problems
  [I probably overuse CP.]

- The computational force of the solvers
  * theory
  * technology
  * global constraints

- The modeling power: high level concepts:
  * global constraints (“Tool for Thought”)
  * element
  * reification

Learning Constraint Programming: How?

When asked how to learn CP I tend to answer something like this:

- Are there any requirements/preferences for using a host language?
  Then pick a system using that language as a host language.

- Are there any special requirements, e.g. need of special constraints
  Then one should probably go for a system that support these
  constraints even though it's not the favourite languages.

  [It's probably not easy for a newcomer in CP to know which
   features that are needed.]

- Is the person new to CP?
  Then consider also learning CP by using very high level systems, e.g.
  MiniZinc, Comet, OPL.

  The high level systems are also excellent for prototyping problems
  before implementing them in “lower level” systems.

Why is CP still relatively unknown?

- Why is CP still unknown?
  * What is it that is unknown? Theory, tools, modeling?
  * Confusion with LP/MIP?
  * Known but considered too specialized?
  * A general view that CP is only CLP?
  * Which myths about CP are there?

- How can CP be more known outside the research field?
  * There are not very much introduction materials/books about CP Modeling, e.g.
    - Helmut Simonis' excellent ECLiPSe videos
    - Pascal Van Hentenryck's Coursera Course “Discrete Optimization”

    I miss a modern introduction book focused on CP modeling. (And I'm not alone.)

- There are some CP blogs/bloggers, but perhaps too few to make an impact. [See later slides for a list.]

- What is the role of CP solver developers in all this?

Role of CP solver developers

The CP system/solver is probably the first contact with CP for a new user.
This is an important role.

What can system developers do?

- Good documentation
- Many examples [You are most welcome to use any of my models.]
- If possible: as easy as possible to model a problem
  This means syntax! Or rather: common tasks should be easy to do, e.g element, reification, loops, etc.

- Start a blog
  * a blog is more public than a forum (and more search engine friendly)
  * features are described, questions are asked
  * e.g. AIMMS blog: http://blog.aimms.com/
  * tweet/Google+/Facebook/etc about it
  * make noise: relevant noise

- Be active on popular programming sites such as Stack Overflow.

Comparison of systems, a subjective feature matrix

- The next slides contains some feature matrices for some CP systems.

- What I've focused on is how easy it is to learn (general) CP using these systems

- Please note that this is a subjective feature mix for learning CP using
  a specific system.
  The criteria mix is subjective as well. What is important when learning CP?

- The matrix only includes systems that I have tested.
  There are many other systems. See the CP Solver Catalog (earlier slide)

- Disclaimer:
  When actually purchasing a system there are many other requirements
  involved, especially commercial systems:
  * professional support
  * specific constraints for the current problem area, e.g. scheduling
  * price
  * reliability of vendor
  * support for a specific host language
  * etc.

Comparison of systems, a subjective feature matrix

Criteria:

- # my models: the number of my (published) models (approx)
- ease of modelling: 5: very easy (=high level), 1: not easy
- documentation, site: how much documentation in distribution or the site
- num. examples: number of examples in the distribution (more is better)
- num. constraints: number of constraints (more is better)
- active community: 5: very active community, 1: no community
- input/output: ease of communicating with external world (files)
- command line option: ease of handling parameters from the user
- reification: 5: very easy/intuitive to handle reification, 1: no reification
- propagators etc: ease of writing propagators or similar constructs
- element: ease of writing element constraints. “x = y[z]” is a 5.
- set var: supports set variables
- debugging: ease (or possibility) to debug a CP model
- CVS/SVN/GitHub/etc: public available source code
- FlatZinc: supports a FlatZinc solver
- open source: is open source project (“free”, under some license)
- commercial: is a commercial product
- host language: host language, if applicable
- latest release (when): when was the latest release (checked in September 2013)

Comparison of the systems, subjective feature matrix

Range: 1..5 where 5 is very good, 1 is not good (or N/A).

	MiniZinc	Comet	Choco2	JaCoP	Gecode	Gecode/R
- # my models	1046	168	21	18	165	29
- ease of modelling	5	5	3	3	4	4
- documentation, site	3	4	4	4	4	3
- num. examples	4	4	3	4	4	3
- num. constraints	4	4	4	4	4	3
- active community	2	2	3	3	4	1
- input/output	2	3	4	4	5	4
- command line option	2	4	4	4	5	3
- reification	5	5	3	3	4	3
- propagators etc	1	4	4	4	4	2
- element	5	5	4	3	4	3
- set var	5	5	5	5	5	4
- debugging	2	5	3	3	4	3
- CVS/SVN/GitHub/etc	1	1	4	5	5	5
- FlatZinc	5	1	3	5	5	1
- open source	3	1	5	5	5	5
- commercial	N	Y	N	N	N	N
- host language	-	-	Java	Java	C++	Ruby
- latest release (when)	201209	2010	201208	201204	201306	2009?

Note: These are features (and my subjected grades) for ease of **learning/modeling**.

Comparison of the systems, subjective feature matrix

Range: 1..5 where 5 is very good, 1 is not good (or N/A).

	ECLiPSe	Essence'	JaCoP/ Scala	SICStus	Zinc	AnsSet Prog
- # my models	178	56	39	152	39	87
- ease of modelling	4	4	4	4	5	4
- documentation, site	5	3	2	5	4	4
- num. examples	4	3	3	5	3	3
- num. constraints	4	3	4	4	4	2
- active community	4	1	2	4	1	4
- input/output	4	2	4	4	2	2
- command line option	3	2	4	4	2	2
- reification	4	5	4	4	5	3
- propagators etc	3	2	3	4	1	1
- element	4	4	4	4	5	2
- set var	5	1	1	4	5	4
- debugging	4	2	3	4	2	2
- CVS/SVN/GitHub/etc	5	1	4	1	1	4
- FlatZinc	5	1	1	4	5	1
- open source	5	3	5	1	4	5
- commercial	N	N	N	(N)	N	N
- host language	Prolog	-	Scala	Prolog	-	-
- latest release	201308	201208	201204	201210	201201	201306

Note: These are features (and my subjected grades) for ease of **learning/modeling**.

Comparison of the systems, subjective feature matrix

Range: 1..5 where 5 is very good, 1 is not good (or N/A).

	or-tools	Python	Java	C#
- # my models	202	36	129
- ease of modelling	4	3	4
- documentation, site	3-4	3-4	3-4
- num. examples	4	3	4
- num. constraints	4	4	4
- active community	4	4	4
- input/output	4	4	4
- command line option	5	5	5
- reification	4	3	4
- propagators etc	4	4	4
- element	4	3	4
- set var	1	1	1
- debugging	4	4	4
- CVS/SVN/GitHub/etc	5	5	5
- FlatZinc	1	1	1
- open source	5	5	5
- commercial	N	N	N
- host language	Python	Java	C#
- last release	201307	201307	201307

Note: These are features (and my subjected grades) for ease of **learning/modeling**.

Comparison of the systems, subjective feature matrix

Range: 1..5 where 5 is very good, 1 is not good (or N/A).

	OscaR	JSR-331	Numberjack
- # my models	147	207	90
- ease of modelling	4	4	4
- documentation, site	3	4	3
- num. Examples	4	3	3
- num. Constraints	4	4	4
- active community	4	3	2
- input/output	4	4	4
- command line option	4	4	4
- reification	4	3	4
- propagators etc	3	3	3
- element	4	3	4
- set var	2	3	-
- debugging	4	3	4
- CVS/SVN/GitHub/etc	5	1	5
- FlatZinc	2	1	2
- open source	5	3	5
- commercial	N	N	N
- host language	Scala	Java	Python
- last release	201308	201204	201308

Note: These are features (and my subjected grades) for ease of **learning/modeling**.

Comparison of the systems, subjective feature matrix

Range: 1..5 where 5 is very good, 1 is not good (or N/A).

	B-Prolog	Choco3	Picat
- # my models	207	90	260
- ease of modelling	4	4	4
- documentation, site	4	3	4
- num. Examples	3	3	3
- num. Constraints	3	4	3
- active community	2	3	2
- input/output	2	4	2
- command line option	2	4	2
- reification	3	3	3
- propagators etc	3	4	2
- element	4	3	3
- set var	1	3	2
- debugging	3	3	3
- CVS/SVN/GitHub/etc	1	5	1
- FlatZinc	3	3	4
- open source	1	5	3
- commercial	N	N	N
- host language	Prolog	Java	-
- last release	201204	201308	201308

Note: These are features (and my subjected grades) for ease of **learning/modeling**.

Comparison of the systems, subjective feature matrix

Range: 1..5 where 5 is very good, 1 is not good (or N/A).

	AIMMS-CP	AMPL+CP	ILOG OPL
- # my models	111	124	117
- ease of modelling	3-5	4	5
- documentation, site	5	4	5
- num. Examples	4	5	4
- num. Constraints	4	3	4
- active community	5	5	5
- input/output	4	4	4
- command line option	2	3	3
- reification	4	3	5
- propagators etc	3	3	3
- element	4	2	4
- set var	3	4	4
- debugging	3	4	4
- CVS/SVN/GitHub/etc	1	(1-5)	1
- FlatZinc	1	1	(4?)
- open source	1	(1-5)	1
- commercial	Y	Y	Y
- host language	-	-	-
- last release	201204	201308	201306

Note: These are features (and my subjected grades) for ease of **learning/modeling**.

FlatZinc solvers that I use (some more than other)

	Approach	Source	Sols	Language	Set	Strength	Heuristics
- G12/FD	CP	N	N/All	Mercury	Y	3-4	Y
- G12/LazyFD	Lazy	N	N/All	Mercury	Y	4-5	Y/N
- G12/CPX	Lazy	N	1/All	C++	Y	3-4	Y
- G12/CBC	MIP	N	1	Mercury?	N	2	?
- Opturion CPX2	CP+SAT	N	N/All	Mercury?	Y	4	Y
- Gecode	CP	Y	N/All	C++	Y	5	Y
- JaCoP	CP	Y	N/All	Java	Y	4-5	Y
- SICStus	CP	Y	N/All	Prolog	N	4	Y
- ECLiPSe	CP/MIP	Y	N/All	Prolog	Y	3-4	Y
- or-tools	CP	Y	N/All	C++	N	5	Y
- fzn2smt	SMT	N	1	C++/Java	Y	4	N
- fzntini	SAT	N	1	C++(?)	Y?	3-4	N
- Chuffed	Lazy	N	N/All	C++	N	4-5	Y
- BProlog	CP	Y	N?/All	Prolog	Y?	2-3	Y?
- minicsp	CP/SAT	N	1	?	Y	4	Y
- minisatid	SAT	Y	1	C++?	Y	3-4	N
- Mistral	CP/SAT	Y	1/all	C++	Y	3-4	Y
- Numberjack	CP/SAT/MIP	Y	1	Python	N?	3	Y
- Choco	CP	Y	N?/All	Java	Y	2-3	Y
- SCIP	MIP	Y	1	C++	N	1-2	?
- Picat	CP/SAT	N	N/All	Picat	N	3	Y
- izplus	CP/SAT	N	N/All	?	Y	3-4	N

Forthcoming tests (perhaps)

- Some of these CP system might be tested in the future:

  * Objective-CP (Objective-C), presented at this conference
  * Copris, http://bach.istc.kobe-u.ac.jp/copris/ (Scala)
  * JaCoP v 4.0 (Java)
  * CloCoP (Clojure wrapper for JaCoP)
  * Clojure core.logic, https://github.com/clojure/core.logic
  * Monadic CP (Haskell)
  * or-tools/C++ (no syntactic sugar at all...)
  * Answer Set Programming, clingcon (Potassco)
  * any new FlatZinc solver

- Other suggestions?
  I'm happy to get other tips (hakank@gmail.com).
  Preferable in Linux with a not too restrictive time/space limitation

CP bloggers

There are few CP bloggers (tweeters etc) compared to the OR bloggers:

- Jean-Charles Regin/Pierre Schaus: "CP is fun"
  http://cp-is-fun.blogspot.com/

- Jacob Feldman: "CP Standardization Blog"
  http://cpstandard.wordpress.com/

- Helmut Simonis: "CP Applications Blog"
  http://hsimonis.wordpress.com/

- Hakan Kjellerstrand: “My Constraint Programming Blog”
  http://www.hakank.org/constraint_programming_blog/

- Thiago Serra: “Thiago Serra's Blog”
  http://thiagoserra.com/

OR people that sometimes blog about CP

- Mike Trick: "Michael Trick's Operations Research Blog"
  http://mat.gsia.cmu.edu/blog/

- Jean-Francois Puget: "IT Best Kept Secret Is Optimization"
  https://www.ibm.com/developerworks/community/blogs/jfp/?lang=en

- Erwin Kalvelagen: "Yet Another Math Programming Consultant"
  http://yetanothermathprogrammingconsultant.blogspot.com/

- Paul Rubin: "OR in an OB World"
  http://orinanobworld.blogspot.com/

- Tallys Yunes: "O.R. By the Beach"
  http://orbythebeach.wordpress.com/

- Stefano Gualandi: "Spaghetti Optimization"
  http://stegua.github.io/

- AIMMS: "AIMMS Blog"
  http://blog.aimms.com/

Thank you!

- Questions?
- Comments?

Håkan Kjellerstrand (hakank@gmail.com)
http://www.hakank.org/
http://www.hakank.org/constraint_programming_blog/

"CP Solvers/Learning Constraint Programming"
CP 2013, 2013-09-16, Uppsala, Sweden
Workshop
"CP Solvers: Modeling, Applications, Integration, and Standardization"