CSPs and Quantified CSPs

G. Verger

LIRMM - CNRS/University of Montpellier II, France

June 19th 2008 / SWIM 2008
What is as CSP?

- Consist of:
  - A set of variables \( X = \{x_1, \ldots, x_n\} \)
  - Variables’ domains (finite sets of possible values)
  - A set of constraints \( C = \{c_1, \ldots, c_k\} \)

**Example**

\[ x \in \{1, 2\}, y \in \{1, 2\}, z \in \{1, 2\}, x = y, x \neq z, y > z \]

- Solutions of a CSP:
  - Assignment of value from its domain to every variable satisfying all the constraints.

**Example**

\((x, y, z) = (2, 2, 1)\) is a solution of above problem
Systematic Search Methods

Characteristics

- Exploring the search space
- Complete and sound
- Efficiency issues

Two algorithms

- Generate-and-Test (GT)
- Backtracking (BT)
Generate-and-Test

- Probably the most general problem solving method

Algorithm:
1. Instantiate all variables
2. Test its satisfaction
3. If not a solution, go back to 1

Drawbacks:
- Blind generator

Possible improvements
- Guiding the generator (⇒ Local Search)
- Checking consistency during generation (⇒ BT)
Backtracking

- Incrementally extends a partial solution towards a complete solution.

Algorithm:
1. Instantiate a variable
2. Check the consistency
3. If there is a variable not instantiated, go to 1

Drawbacks:
- Trashing, redundant work
- Late detection of inconsistencies

Possible improvements
- Loop-back methods
- Look-ahead methods
CSP as a graph:
- Nodes are variables
- Edges are constraints

Aim: Removing inconsistent values from domains

Different kinds of techniques:
- Node Consistency (NC)
- Arc Consistency (AC)
- Path Consistency (PC)
- k-consistency

Not Complete
Constraint Propagation

- Systematic search = not efficient
- Consistency checking = not complete
- Combine search and check!
- Two methods:
  - Look-ahead: prevent conflicts (FC, AC, SAC...)
  - Look-back: intelligent restoring after conflict (Backjumping, Dynamic BT)
- Current Solvers use BT + AC.
Heuristics

- CSP is NP-Complete
- Choices are made during BT
- ⇒ are there better ways to explore the tree search?
- Two kinds of heuristics:
  - Variable Ordering Heuristics = Fail-First (MinDom, MaxDeg, Dom/Deg, Dom/wDeg)
  - Value Ordering Heuristics = Succeed-First (MinConflict, Geelen’s Promise)
- Current Solvers use Dom/wDeg + MinConflict
What is a QCSP

Consists of:
- A sequence of variables $X = \{x_1, \ldots, x_n\}$
- variables’ quantifiers (existential or universal)
- variables’ domains (finite sets of possible values)
- A set of constraints $C = \{c_1, \ldots, c_k\}$

Example

$$\exists x_1 \in \{1, 2, 3\} \forall y_1 \in \{1, 2\} \exists x_2 \in \{1, 2\} \ x_1 \neq x_2, y_1 = x_2$$
Definition of a QCSP (2)

Example

\[ \exists x_1 \in \{1, 2, 3\} \land y_1 \in \{1, 2\} \exists x_2 \in \{1, 2\} \quad x_1 \neq x_2, y_1 = x_2 \]

- A Scenario is:
  - an assignment of value from its domain to every variable satisfying all the constraints.
  - the solution of corresponding CSP

Example

\((x_1, y_1, x_2) = (1, 2, 2)\) is a scenario of the previous QCSP
Definition of a QCSP (3)

Example

$$\exists x_1 \in \{1, 2, 3\} \forall y_1 \in \{1, 2\} \exists x_2 \in \{1, 2\} \ x_1 \neq x_2, y_1 = x_2$$

- A Strategy is:
  - the tree of winning scenarii corresponding to the QCSP
Solving a QCSP

- Transforming into CSP: exponential space!
- what techniques can we bring from classical CSP?
  - Backtracking?
  - Constraint Checking?
  - Heuristics?
Backtracking seems a reasonable start

**Algorithm:**
- Instantiate a variable
- Check the consistency
- If there is a variable not instantiated, go to 1
- If all are instantiated, go to last unchecked universal variable
Backtracking seems a reasonable start

Algorithm:

- Instantiate a variable *(Order of variables!)*
- Check the consistency
- If there is a variable not instantiated, go to 1
  - if all are instantiated, go to last unchecked universal variable
Backtracking seems a reasonable start

**Algorithm:**

- Instantiate a variable *(Order of variables!)*
- Check the consistency *(Adapt Consistency techniques)*
- If there is a variable not instantiated, go to 1
- If all are instantiated, go to last unchecked universal variable
Backtracking seems a reasonable start

**Algorithm:**

- Instantiate a variable *(Order of variables!)*
- Check the consistency *(Adapt Consistency techniques)*
- If there is a variable not instantiated, go to 1
- If all are instantiated, go to last unchecked universal variable
Consistency Techniques

- As in CSP for existential variables
- for universal variables: removing one value = Fail

The example of Quantified AC

\[ x \in \{1, 2, 3\}, y \in \{1, 2\} \]

- \( \exists x \exists y, x \neq y \) AC detects nothing
- \( \forall x \exists y, x \neq y \) AC detects nothing
- \( \forall x \forall y, x \neq y \) AC detects inconsistency!
- \( \exists x \forall y, x \neq y \) AC removes 1 and 2 from \( x \)'s domain!

\( \forall x \forall y \) and \( \exists x \forall y \) can be checked during preprocessing!
Heuristics

- QCSP is PSPACE-Complete
- Choices are made during BT
  - ⇒ are there better ways to explore the tree search?
- Two kinds of heuristics:
  - Variable Ordering Heuristics
    - Fail-First (MinDom, MaxDeg, Dom/Deg, Dom/wDeg)
    - Inside a block!
  - Value Ordering Heuristics
    - Succeed-First (MinConflict) ⇒ Existential variables
    - Fail-First (MaxConflict) ⇒ Universal variables
Some techniques from boolean world.

- **The Pure-Value Rule**
  - From Pure-Literal in QBF
- **Solution-Directed BackJumping**
  - Cube learning in QBF
Constraint Satisfaction Problem:
- Since 1974 (Montanari), 1977 (Mackworth)
- Solvers: ILOG Solver, JChoco, Minion...
- Benchmarks: CSP-lib on the web

Quantified CSP:
- Since 2002 (Bordeaux, Monfroy)
- No reference
- Solvers: Qcsp-solve, QeCode, BlockSolve
- No benchmarks!