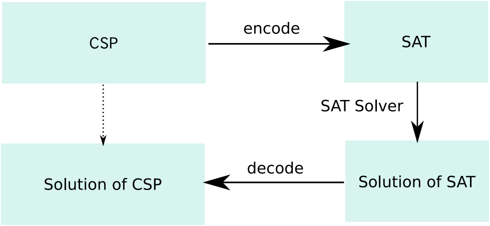
Sugar: A SAT-based CSP Solver

Naoyuki Tamura, Tomoya Tanjo, and Mutsunori Banbara

Kobe University, Japan

Sep. 14, 2008

Features of Sugar CSP Solver



- Sugar is a SAT-based solver for CSPs, COPs, and Max-CSPs.
- It uses order encoding method (Tamura et al. CP2006) which is better for various problems than other encodings, such as direct encoding and support encoding.
- SAT problems are solved by an external efficient SAT solver, such as MiniSat and PicoSAT.

Order encoding

- Order encoding is a generalization of the encoding method originally used by Crawford and Baker for Job-Shop Scheduling problems.
- It uses a different Boolean variable $P_{x,a}$ representing $x \le a$ for each integer variable x and integer value a.

$$P_{x,a} \iff x \le a$$

Order encoding

- Order encoding is a generalization of the encoding method originally used by Crawford and Baker for Job-Shop Scheduling problems.
- It uses a different Boolean variable $P_{x,a}$ representing $x \le a$ for each integer variable x and integer value a.

$$P_{x,a} \iff x \le a$$

- For example, the following four Boolean variables are used to encode an integer variable $x \in \{1, 2, 3, 4, 5\}$.

$$P_{x,1} \qquad P_{x,2} \qquad P_{x,3} \qquad P_{x,4}$$

Please note $P_{x,5}$ (i.e. $x \le 5$) is not necessary because it is always true.

Order encoding of variables

- **Integer variable** $x \in \{1, 2, 3, 4, 5\}$ can be encoded into the following *three* SAT clauses while the direct encoding requires 11 clauses (one at-least-one clause and 10 at-most-one clauses).

$$
\begin{array}{ll}
\neg P_{x,1} \lor P_{x,2} & \text{(i.e. } (x \le 1) \supset (x \le 2)\text{)} \\
\neg P_{x,2} \lor P_{x,3} & \text{(i.e. } (x \le 2) \supset (x \le 3)\text{)} \\
\neg P_{x,3} \lor P_{x,4} & \text{(i.e. } (x \le 3) \supset (x \le 4)\text{)}
\end{array}
$$

Order encoding of variables

- **Integer variable** $x \in \{1, 2, 3, 4, 5\}$ can be encoded into the following *three* SAT clauses while the direct encoding requires 11 clauses (one at-least-one clause and 10 at-most-one clauses).

$$
\begin{array}{ll}
\neg P_{x,1} \lor P_{x,2} & (\text{i.e. } (x \le 1) \supset (x \le 2)) \\
\neg P_{x,2} \lor P_{x,3} & (\text{i.e. } (x \le 2) \supset (x \le 3)) \\
\neg P_{x,3} \lor P_{x,4} & (\text{i.e. } (x \le 3) \supset (x \le 4))
\end{array}
$$

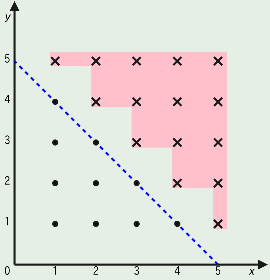
- The followings are the satisfiable assignments.

$P_{x,1}$	$P_{x,2}$	$P_{x,3}$	$P_{x,4}$	Interpretation
T	T	T	T	$x = 1$
F	T	T	T	$x = 2$
F	F	T	T	$x = 3$
F	F	F	T	$x = 4$
F	F	F	F	$x = 5$

Order encoding of linear constraints

- **Constraints** can be encoded by representing conflict regions instead of conflict points as used in direct encoding.
- For example, $x + y \leq 5$ is encoded into the following *five* SAT clauses while the direct encoding requires 15 clauses.

Clauses	Conflict regions
$P_{y,4}$	$\neg((x \geq 1) \land (y \geq 5))$
$P_{x,1} \lor P_{y,3}$	$\neg((x \geq 2) \land (y \geq 4))$
$P_{x,2} \lor P_{y,2}$	$\neg((x \geq 3) \land (y \geq 3))$
$P_{x,3} \lor P_{y,1}$	$\neg((x \geq 4) \land (y \geq 2))$
$P_{x,4}$	$\neg((x \geq 5) \land (y \geq 1))$



Order encoding of other intensional constraints

Expression	Replacement	Extra condition
$E < F$	$E+1 \le F$	
$E = F$	$(E \le F) \wedge (E \ge F)$	
$E \ne F$	$(E < F) \vee (E > F)$	
$\max(E, F)$	x	$(x \ge E) \wedge (x \ge F) \wedge ((x \le E) \vee (x \le F))$
$\min(E, F)$	x	$(x \le E) \wedge (x \le F) \wedge ((x \ge E) \vee (x \ge F))$
$\mathrm{abs}(E)$	x	$(x \ge E) \wedge (x \ge -E) \wedge ((x \le E) \vee (x \ge -E))$
E div c	q	$(E = c\,q + r) \wedge (0 \le r) \wedge (r < c)$
E mod c	r	$(E = c\,q + r) \wedge (0 \le r) \wedge (r < c)$

- Other intensional expressions are translated as described above.

Order encoding of other intensional constraints

Expression	Replacement	Extra condition
$E < F$	$E + 1 \leq F$	
$E = F$	$(E \leq F) \wedge (E \geq F)$	
$E \neq F$	$(E < F) \vee (E > F)$	
$\max(E, F)$	x	$(x \geq E) \wedge (x \geq F) \wedge ((x \leq E) \vee (x \leq F))$
$\min(E, F)$	x	$(x \leq E) \wedge (x \leq F) \wedge ((x \geq E) \vee (x \geq F))$
$\mathrm{abs}(E)$	x	$(x \geq E) \wedge (x \geq -E) \wedge ((x \leq E) \vee (x \geq -E))$
$E \text{ div } c$	q	$(E = c\,q + r) \wedge (0 \leq r) \wedge (r < c)$
$E \text{ mod } c$	r	$(E = c\,q + r) \wedge (0 \leq r) \wedge (r < c)$

- Other intensional expressions are translated as described above.
- Encodings of variable multiplications (such as $x \cdot y$) are possible, but the current Sugar implementation does not support them.
- However, some heuristic conversion rules (e.g. $x \cdot y > 0$) are introduced to reduce the demerits.

Order encoding of global constraints

- alldifferent$(x_1, x_2, \ldots, x_n)$ constraint is translated as follows:

$$\bigwedge_{i<j}(x_i \neq x_j)$$
$$\neg \bigwedge(x_i < lb + n - 1)$$
$$\neg \bigwedge(x_i > ub - n + 1)$$

  where the last two are extra pigeon hole constraints, and lb and ub are the lower and upper bounds of $\{x_1, x_2, \ldots, x_n\}$.
- Other global constraints are translated in a straightforward way.

Order encoding of extensional constraints

- **Conflict tuples** are combined into conflict regions and encoded by the order encoding.
- **Support tuples** are encoded by considering their complements.

Solving Max-CSPs and COPs

- **Max-CSP** is solved by converting it into an equivalent COP.
- **COP** is solved by repeatedly invoking a SAT solver with varying the bound condition of the objective variable in a bisection-search way.

Solving Max-CSPs and COPs

- **Max-CSP** is solved by converting it into an equivalent COP.
- **COP** is solved by repeatedly invoking a SAT solver with varying the bound condition of the objective variable in a bisection-search way.
- Detailed explanations will be given in the next talk.