Encoding High-Level Constraints into SAT and MIP

Neng-Fa Zhou

Brooklyn College & Graduate Center
The City University of New York

PicatSAT's Performance at 2022 FLOC Olympic Games

PicatSAT's Performance in 2024

- **XCSP Competition**

Rank	Main CSP	Main COP	Fast COP
1st	Picat	CPMpy_ortools	Picat
2nd	CPMpy-ortools	Picat	CoSoCo
3rd	Fun-sCOP	CoSoCo	Choco

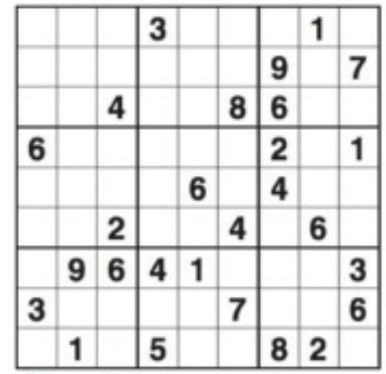
- **MiniZinc Challenge**

Category	Gold	Silver	Bronze
Fixed	OR-Tools CP-SAT	Choco-solver CP-SAT	SICStus Prolog
Free	OR-Tools CP-SAT	PicatSAT	iZplus
Parallel	OR-Tools CP-SAT	PicatSAT	Choco-solver CP
Open	OR-Tools CP-SAT	PicatSAT	Choco-solver CP
Local Search	OR-Tools CP-SAT LS	Yuck	

```
import sat.

sudoku(Board) =>
   N = Board.len,
   Vars = Board.vars(),
   Vars :: 1..N,
   foreach (Row in Board)
      all_different(Row)
   end,
   foreach (J in 1..N)
      all_different([Board[I,J] : I in 1..N])
   end,
   M = round(sqrt(N)),
   foreach (I in 1..M..N-M, J in 1..M..N-M)
      all_different([Board[I+K,J+L] : K in 0..M-1, L in 0..M-1])
   end,
   solve(Vars).
```

cp
mip
smt



Outline

- **Encoding CSP into SAT**
  - *Encodings and Optimizations*
- **Encoding CSP into MIP**
- ***Summary***

The Satisfiability Problem (SAT)

Given a Boolean formula, the SAT problem is to determine if the formula is satisfiable. If yes, it finds an assignment for the variables that makes the formula satisfiable.

$$(x_1 \rightarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4) \wedge \neg x_2$$

$$\downarrow$$

$$\neg x_1 \vee x_2 \vee x_3$$

$$\neg x_1 \vee x_2 \vee \neg x_4$$

CNF

G.S. Tseytin: On the complexity of derivation in propositional calculus, 1969.

SAT Solving (CDCL)

1. *Choice*: Assign a value to a selected variable.
2. *Unit propagation*: Use this assignment to determine values for the other variables.
3. *Backjump*: If a conflict is found, add the negation of the conflict-causing clause as a new clause and backtrack to the choice that made the conflict occur.
4. Continue from step 1.

- Martin Davis, Hilary Putnam: A Computing Procedure for Quantification Theory, 1960.
- Martin Davis, George Logemann, Donald Loveland: A Machine Program for Theorem Proving, 1962.
- Joao Marques-Silva, Ines Lynce and Sharad Malik: Conflict-Driven Clause Learning SAT Solvers, 2009.
- J. K. Fichte, D. Le Berre, M. Hecher, S. Szeider: The Silent (R)evolution of SAT, 2023.

SAT Encodings

$$X :: \{a_1, a_2, \ldots, a_n\}$$

- Direct encoding

$$B_i \leftrightarrow X = a_i$$
$$B_1 \ \vee \ B_2 \ \vee \ \ldots \ \vee \ B_n$$
$$\texttt{at\_most\_one}([B_1, B_2, \ldots, B_n])$$

- Order encoding

$$B_i \leftrightarrow X \leq a_i$$

- Log encoding (sign-and-magnitude)

$$X.\mathrm{m} = \, <B_{k-1} \ldots B_1 B_0>$$
$$X.\mathrm{s} = 0 \text{ or } 1$$

- Johan de Kleer: A Comparison of ATMS and CSP Techniques,1989.
- James M. Crawford, Andrew B. Baker: Experimental Results on the Application of Satisfiability Algorithms to Scheduling Problems, 1994.
- Kazuo Iwama, Shuichi Miyazaki: SAT - Variable Complexity of Hard Combinatorial Problems, 1994.

CSP Solvers based on SAT

- BEE (order encoding)
- FznTini (log encoding)
- meSAT (order and direct encodings)
- PicatSAT (log and direct encodings)
- Savile Row (order and direct encodings)
- Sugar (order encoding) and its successors

- Naoyuki Tamura, Akiko Taga, Satoshi Kitagawa, Mutsunori Banbara: Compiling finite linear CSP into SAT, 2009.
- Jinbo Huang: Universal Booleanization of Constraint Models, 2008.
- Amit Metodi, Michael Codish: Compiling finite domain constraints to SAT with BEE, 2012.
- Mirko Stojadinovic, Filip Maric: meSAT - multiple encodings of CSP to SAT, 2014.
- Neng-Fa Zhou and Hakan Kjellerstrand: Optimizing SAT Encodings for Arithmetic Constraints, 2017.

The PicatSAT Compiler

High-level Constraints

preprocessing

decomposition

Primitive Constraints

- Small Pseudo-Boolean (PB) constraint: $\Sigma_1^n(a_i \times B_i) \odot b$, $n =< 10$
- Boolean cardinality constraint: $\Sigma_i^n B_i \odot b$ (b is 1 or 2)
- $X :: D$
- $X \odot Y$
- $X + Y = Z$
- $X \times Y = Z$
- $Y = \text{-}X$
- $\texttt{abs}(X) = Y$
- $X \texttt{ div } Y = Z$
- $X \texttt{ mod } Y = Z$
- Reified primitive constraint: $B \Leftrightarrow C$
- Implicative primitive constraint: $B \Rightarrow C$

- Constraints are made to be arc-consistent or interval consistent
- No primitive constraints are duplicated
- Avoid creating large-domain variables
- Avoid creating domains with negative values

Encoding Small PB Constraints

- Espresso

```
% X+Y = 2*U+V,
% where [X,Y,U,V] :: 0..1, UV ≠ 11

.i 4
.o 1
0000 0
0001 1
0010 1
0011 -
0100 1
0101 0
0110 1
0111 -
1000 1
1001 0
1010 1
1011 -
1100 1
1101 1
1110 0
1111 -
```

→

```
.i 4
.o 1
.p 5
1-00 1
-01- 1
01-0 1
110- 1
00-1 1
.e
```

$$\neg X \vee U \vee V$$
$$Y \vee \neg U$$
$$X \vee \neg Y \vee V$$
$$\neg X \vee \neg Y \vee U$$
$$X \vee Y \vee \neg V$$

Encoding the at-most-one Constraint

- Chen's two-product encoding

$$
\begin{array}{c|ccccc}
 & V_1 & V_2 & \dots & V_j \;\dots & V_q \\
\hline
U_1 & X_{11} & X_{12} & & \dots & X_{1q} \\
U_2 & X_{21} & X_{22} & & \dots & X_{2q} \\
. & & & & & \\
U_i & & \dots & & X_{ij} & \dots \\
. & & & & & \\
U_p & X_{p1} & X_{p2} & & \dots & X_{pq}
\end{array}
$$

$\boxed{\texttt{amo}([X_{11},\ X_{12},\ \dots,\ X_{pq}])}$

$\downarrow$

$\texttt{amo}([U_1,\ U_2,\ \dots,\ U_p])$

$\texttt{amo}([V_1,\ V_2,\ \dots,\ V_q])$

$X_{ij} \rightarrow U_i$ /\ V_j
for i in 1..p and j in 1..q

Jingchao Chen: A New SAT Encoding of the At-Most-One Constraint, 2010.

Sign-and-Magnitude Log Encoding

X :: D

- Each domain variable is encoded as a vector of Boolean variables
  - $X.m = \langle B_{n-1},\ldots,B_1,B_0 \rangle$
  - X.s is the sign bit
- No negative zero is allowed
  - $X.m = \langle 0,\ldots,0,0 \rangle \Rightarrow X.s = 0$

Sign-and-Magnitude Log Encoding (Example)

```
X :: [-2,-1,1,2]

X.m = <X1,X0>
X.s = S
```

- Naïve Encoding

$$\begin{array}{ll} \neg S \vee \neg X_1 \vee \neg X_0 & (X \neq -3) \\ \neg S \vee X_1 \vee X_0 & (X \neq -0) \\ S \vee X_1 \vee X_0 & (X \neq 0) \\ S \vee \neg X_1 \vee \neg X_0 & (X \neq 3) \end{array}$$

- Optimized Encoding (Using Espresso)

$$\begin{array}{l} \mathbf{X}_0 \vee \mathbf{X}_1 \\ \neg\mathbf{X}_0 \vee \neg\mathbf{X}_1 \end{array}$$

The Comparison Constraint: X ≥ Y

- Signed comparison

$$
\begin{aligned}
&X.\mathrm{s} = 0 \wedge Y.\mathrm{s} = 1 \vee \\
&X.\mathrm{s} = 1 \wedge Y.\mathrm{s} = 1 \Rightarrow X.\mathrm{m} \leq Y.\mathrm{m} \vee \\
&X.\mathrm{s} = 0 \wedge Y.\mathrm{s} = 0 \Rightarrow X.\mathrm{m} \geq Y.\mathrm{m}
\end{aligned}
$$

- Unsigned comparison

$$X.\mathrm{m} = <X_{n-1}X_{n-2}\ldots X_1X_0>,\ Y.\mathrm{m} = <Y_{n-1}Y_{n-2}\ldots Y_1Y_0>$$

$$
\begin{aligned}
&T_0 \Leftrightarrow (X_0 \geq Y_0) \\
&T_1 \Leftrightarrow (X_1 > Y_1) \vee (X_1 = Y_1 \wedge T_0) \\
&\qquad\qquad \vdots \\
&T_{n-1} \Leftrightarrow (X_{n-1} > Y_{n-1}) \vee (X_{n-1} = Y_{n-1} \wedge T_{n-2})
\end{aligned}
$$

The Addition Constraint: X+Y = Z

- Unsigned addition (ripple-carry adders)

$$
\begin{array}{r}
X_{n-1} \ldots X_1\ X_0 \\
+\ \ Y_{n-1} \ldots Y_1\ \ Y_0 \\
\hline
Z_n\ Z_{n-1} \ldots Z_1\ \ Z_0
\end{array}
$$

Carriers are used

- Signed addition

$$
\begin{array}{l}
X.\mathrm{s} = 0 \wedge Y.\mathrm{s} = 0 \Rightarrow Z.\mathrm{s} = 0 \wedge X.\mathrm{m}+Y.\mathrm{m} = Z.\mathrm{m} \\
X.\mathrm{s} = 1 \wedge Y.\mathrm{s} = 1 \Rightarrow Z.\mathrm{s} = 1 \wedge X.\mathrm{m}+Y.\mathrm{m} = Z.\mathrm{m} \\
X.\mathrm{s} = 0 \wedge Y.\mathrm{s} = 1 \wedge Z.\mathrm{s} = 1 \Rightarrow X.\mathrm{m}+Z.\mathrm{m} = Y.\mathrm{m} \\
X.\mathrm{s} = 0 \wedge Y.\mathrm{s} = 1 \wedge Z.\mathrm{s} = 0 \Rightarrow Y.\mathrm{m}+Z.\mathrm{m} = X.\mathrm{m} \\
X.\mathrm{s} = 1 \wedge Y.\mathrm{s} = 0 \wedge Z.\mathrm{s} = 0 \Rightarrow X.\mathrm{m}+Z.\mathrm{m} = Y.\mathrm{m} \\
X.\mathrm{s} = 1 \wedge Y.\mathrm{s} = 0 \wedge Z.\mathrm{s} = 1 \Rightarrow Y.\mathrm{m}+Z.\mathrm{m} = X.\mathrm{m}
\end{array}
$$

The Full Adder

$$X_i + Y_i + C_{in} = C_{out} Z_i$$

$$X_i \vee \neg Y_i \vee C_{in} \vee Z_i$$
$$X_i \vee Y_i \vee \neg C_{in} \vee Z_i$$
$$\neg X_i \vee \neg Y_i \vee C_{in} \vee \neg Z_i$$
$$\neg X_i \vee Y_i \vee \neg C_{in} \vee \neg Z_i$$
$$\neg X_i \vee C_{out} \vee Z_i$$
$$X_i \vee \neg C_{out} \vee \neg Z_i$$
$$\neg Y_i \vee \neg C_{in} \vee C_{out}$$
$$Y_i \vee C_{in} \vee \neg C_{out}$$
$$\neg X_i \vee \neg Y_i \vee \neg C_{in} \vee Z_i$$
$$X_i \vee Y_i \vee C_{in} \vee \neg Z_i$$

An Optimized Carrier-free Encoding for Y = X+1

$$\begin{array}{rr} & X_{m-1} \ldots X_1\ X_0 \\ + & 1 \\ \hline & Y_{m-1}\ \ldots\ Y_1\ Y_0 \end{array}$$

- Consider two bits a time

$$\neg Y_{i-1} \wedge X_{i-1} \Rightarrow Y_i = \neg X_i$$
$$\neg Y_{i-1} \wedge X_{i-1} \wedge X_i \Rightarrow Y_{i+1} = \neg X_{i+1}$$
$$\text{otherwise} \Rightarrow Y_i = X_i \wedge Y_{i+1} = X_{i+1}$$

→ 11 clauses

- Top-most 4 bits → 21 clauses

$X_{m-1} \vee X_{m-4} \vee \neg Y_{m-1}$
$X_{m-1} \vee \neg Y_{m-1} \vee \neg Y_{m-2}$
$X_{m-1} \vee \neg Y_{m-1} \vee \neg Y_{m-3}$
$X_{m-1} \vee \neg Y_{m-1} \vee \neg Y_{m-4}$
$X_{m-3} \vee \neg Y_{m-3} \vee \neg Y_{m-4}$
$\neg X_{m-4} \vee \neg Y_{m-5} \vee Y_{m-4}$
$X_{m-2} \vee \neg X_{m-3} \vee Y_{m-2} \vee Y_{m-3}$
$X_{m-4} \vee X_{m-5} \vee \neg Y_{m-4}$
$\neg X_{m-1} \vee Y_{m-1}$
$\neg X_{m-2} \vee \neg X_{m-3} \vee \neg Y_{m-2} \vee Y_{m-3}$
$\neg X_{m-3} \vee \neg X_{m-4} \vee \neg X_{m-5} \vee Y_{m-5} \vee \neg Y_{m-3}$

$X_{m-2} \vee X_{m-4} \vee \neg Y_{m-2}$
$X_{m-3} \vee X_{m-4} \vee \neg Y_{m-3}$
$X_{m-2} \vee \neg Y_{m-2} \vee \neg Y_{m-3}$
$X_{m-2} \vee \neg Y_{m-2} \vee \neg Y_{m-4}$
$\neg X_{m-4} \vee X_{m-5} \vee Y_{m-4}$
$X_{m-1} \vee \neg X_{m-2} \vee Y_{m-1} \vee Y_{m-2}$
$X_{m-4} \vee \neg X_{m-5} \vee Y_{m-5} \vee Y_{m-4}$
$X_{m-4} \vee \neg Y_{m-5} \vee \neg Y_{m-4}$
$\neg X_{m-1} \vee \neg X_{m-2} \vee \neg Y_{m-1} \vee Y_{m-2}$
$X_{m-3} \vee \neg X_{m-4} \vee \neg X_{m-5} \vee Y_{m-5} \vee Y_{m-3}$

The Multiplication Constraint: X*Y = Z

- The Shift-and-Add Algorithm

X.m*Y.m = Z.m X.m = <X_{n-1},…,X_1,X_0>

$$
\begin{aligned}
&X_0 = 0 \Rightarrow S_0 = 0\\
&X_0 = 1 \Rightarrow S_0 = Y\\
&X_1 = 0 \Rightarrow S_1 = S_0\\
&X_1 = 1 \Rightarrow S_1 = (Y << 1) + S_0\\
&\vdots\\
&X_i = 0 \Rightarrow S_i = S_{i-1}\\
&X_i = 1 \Rightarrow S_i = (Y << i) + S_{i-1}\\
&\vdots\\
&X_{n-1} = 0 \Rightarrow S_{n-1} = S_{n-2}\\
&X_{n-1} = 1 \Rightarrow S_{n-1} = (Y << (n-1)) + S_{n-2}\\
&Z = S_{n-1}
\end{aligned}
$$

- H. Bierlee, etc.: Single Constant Multiplication for SAT. CPAIOR'24.

Equivalence Reasoning

- Equivalence reasoning is an optimization that reasons about a possible value for a Boolean variable or the relationship between two Boolean variables at compile time.

$$
\begin{array}{lll}
X & = \mathrm{abs}(Y) & \Rightarrow X.\mathrm{m} = Y.\mathrm{m},\ X.\mathrm{s} = 0 \\
X & = \text{-}Y & \Rightarrow X.\mathrm{m} = Y.\mathrm{m},\ X.\mathrm{s} = Y.\mathrm{s} = 0 \to X.\mathrm{m} = 0 \\
X & = Y \bmod 2^K & \Rightarrow X_0 = Y_0,\ X_1 = Y_1,\ \ldots,\ X_{k-1} = Y_{k-1} \\
X & = Y \ \mathrm{div}\ 2^K & \Rightarrow X_0 = Y_K,\ X_1 = Y_{K+1},\ \ldots
\end{array}
$$

- No clauses are needed to encode X = abs(Y).

Constant Propagation on X+Y = Z

$$\boxed{X_i + Y_i = C_{out} Z_i}$$

$$
\begin{aligned}
&\textbf{Rule-1} : X_i = 0 \Rightarrow C_{out} = 0 \wedge Z_i = Y_i.\\
&\textbf{Rule-2} : X_i = 1 \Rightarrow C_{out} = Y_i \wedge Z_i = \neg Y_i.\\
&\textbf{Rule-3} : Z_i = 0 \Rightarrow C_{out} = X_i \wedge X_i = Y_i\\
&\textbf{Rule-4} : Z_i = 1 \Rightarrow C_{out} = 0 \wedge X_i = \neg Y_i.
\end{aligned}
$$

- Example-1

$$
\begin{array}{cccc}
 & X_2 & X_1 & X_0 \\
+ & 1 & 0 & 0 \\
\hline
Z_3 & Z_2 & Z_1 & Z_0
\end{array}
\quad \Rightarrow \quad
\begin{aligned}
X_0 &= Z_0\\
X_1 &= Z_1\\
\neg X_2 &= Z_2\\
X_2 &= Z_3
\end{aligned}
$$

- Example-2

$$
\begin{array}{cccc}
 & X_2 & X_1 & X_0 \\
+ & Y_2 & Y_1 & Y_0 \\
\hline
1 & 0 & 1 & 1
\end{array}
\quad \Rightarrow \quad
\begin{aligned}
\neg X_0 &= Y_0\\
\neg X_1 &= Y_1\\
X_2 &= Y_2\\
X_2 &= 1\\
Y_2 &= 1
\end{aligned}
$$

Constant Propagation on X*Y = Z

$$
\begin{aligned}
&X_0 = 0 \Rightarrow S_0 = 0\\
&X_0 = 1 \Rightarrow S_0 = Y\\
&X_1 = 0 \Rightarrow S_1 = S_0\\
&X_1 = 1 \Rightarrow S_1 = (Y << 1) + S_0\\
&\vdots
\end{aligned}
$$

$$
\begin{aligned}
&X_i = 0 \Rightarrow S_i = S_{i-1}\\
&X_i = 1 \Rightarrow S_i = (Y << i) + S_{i-1}
\end{aligned}
$$

$$
\begin{aligned}
&\vdots\\
&X_{n-1} = 0 \Rightarrow S_{n-1} = S_{n-2}\\
&X_{n-1} = 1 \Rightarrow S_{n-1} = (Y << (n-1)) + S_{n-2}\\
&Z = S_{n-1}
\end{aligned}
$$

Rule 5 : $X_i = 0 \Rightarrow$ copy all of the bits of S_{i-1} into S_i.

Rule 6 : $X_i = 1 \Rightarrow$ copy the lowest i bits of S_{i-1} into S_i.

Rule 7 : $X.m = <X_{n-1} \ldots X_i 0 \ldots 0> \wedge X_i = 1 \quad \Rightarrow$
$Z_i = Y_0 \wedge Z_k = 0$ for $k \in 0..(i-1)$.

all_different(L)

$$\texttt{all\_different}([V_1, V_2, \ldots, V_n])$$

- Standard

  $V_i \neq V_j$ for $i, j = 1, \ldots, n, i < j$.

- Use `at_most_one`
  - Let $D = D_1 \cup D_2 \cup \ldots \cup D_n$
  - If $|D| > n$: $\forall_a \in D$: $\texttt{at\_most\_one}([V_1 = a, V_2 = a, \ldots, V_n = a])$
  - If $|D| = n$: $\forall_a \in D$: $\texttt{exactly\_one}([V_1 = a, V_2 = a, \ldots, V_n = a])$
- A hybrid of log and direct encodings

The cumulative Constraint

$$\texttt{cumulative}([S_1, S_2, \ldots, S_n], [D_1, D_2, \ldots, D_n], [R_1, R_2, \ldots, R_n], Limit)$$

- Occupation constraints

for each time t_i and each task j:

$$O_{ij} \leftrightarrow S_j \leq t_i < S_j + D_j$$

- Resource constraints

for each time t_i:

$$\sum_{j=1}^{n} O_{ij} * R_j \leq Limit$$

- Time points
  - Time decomposition: all the time points in the make span
  - Task decomposition: only the start or end points

- Andreas Schutt, Thibaut Feydy, Peter J. Stuckey, and Mark G.Wallace. Explaining the cumulative propagator, 2011.

The acyclic_d(V, E) Constraint

- Leaf-elimination encoding (LEE)
  - A graph is acyclic if the graph can be reduced to empty after leaves are repeatedly eliminated.

    $$G_0 = G \longrightarrow G_1 \quad \cdots \quad \longrightarrow G_t$$

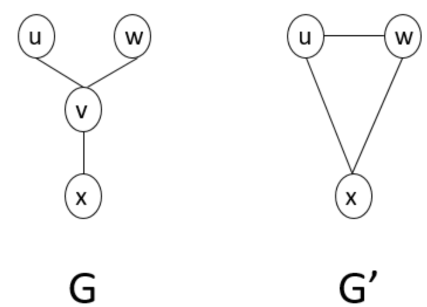
  - Use a time variable for each vertex
- Vertex-elimination encoding (VEE)

  - If G' is cyclic, then G is cyclic
- Hybrid encoding (HYB)
  - Combines LEE and VEE

- M. F. Rankooh and J. Rintanen: Propositional Encodings of Acyclicity and Reachability by Using Vertex Elimination, AAAI'22.
- N.F. Zhou, R. Want, and R. Yap: A Comparison of SAT Encodings for Acyclicity of Directed Graphs, SAT'23.

The hcp(V,E) Constraint

- Distance encoding (DIST)
  - Use a distance variable for each vertex
  - If an arc (u,v) is in the cycle and v is not the starting vertex, then Dv = Du+1
- Vertex elimination encoding (VEE)
  - Ensure the mapping between H_G and $H_{G'}$



- Hybrid encoding combing DIST and VEE

  - N.F. Zhou: In Pursuit of an Efficient SAT Encoding for the Hamiltonian Cycle Problem, CP 2020.
  - N.F. Zhou: Encoding the Hamiltonian Cycle Problem into SAT Based on Vertex Elimination, CP 2024.

Further Readings

- N.F. Zhou, H. Kjellerstrand: The Picat-SAT Compiler, PADL 2016.
- N.F. Zhou, H. Kjellerstrand: Optimizing SAT Encodings for Arithmetic Constraints, CP 2017.
- R. Bartak, N.F. Zhou, R. Stern, E. Boyarski, and P. Surynek: Modeling and Solving the Multi-Agent Pathfinding Problem in Picat, ICTAI'17.
- N.F. Zhou: In Pursuit of an Efficient SAT Encoding for the Hamiltonian Cycle Problem, CP 2020.
- N.F. Zhou: Modeling and Solving Graph Synthesis Problems Using SAT-Encoded Reachability Constraints in Picat, ICLP 2021.
- N.F. Zhou, R. Want, and R. Yap: A Comparison of SAT Encodings for Acyclicity of Directed Graphs, SAT 2023.
- N.F. Zhou: Encoding the Hamiltonian Cycle Problem into SAT Based on Vertex Elimination, CP 2024.

Encoding CSP into MIP

- MIP for combinatorial search
  - Symplex method
  - LP relaxation
  - Branch-and-bound
  - Cutting planes method
- MIP solvers support nonlinear constraints
- Encoding nonlinear constraints into linear ones is still important

CSP to MIP

High-level Constraints

preprocessing

decomposition

Primitive Constraints

- $X :: D$
- Linear constraints
- $X \neq Y$
- $abs(X) = Y$
- $X \times Y = Z$
- $X\ div\ Y = Z$
- $X\ mod\ Y = Z$
- $max(X, Y) = Z$
- $min(X, Y) = Z$
- $B \leftrightarrow C$
- $B \rightarrow C$
- $\texttt{table\_in}(\{X_1, X_2, \ldots, X_n\},\ T)$

- Constraints are made to be arc-consistent or interval consistent
- No primitive constraints are duplicated

$X :: D$

- Let $D = L_1..U_1 \cup L_2..U_2 \cup \ldots \cup L_m..U_m$
- Introduce a binary variable B_i for each interval $L_i..U_i$
  - $B_i \rightarrow X \geq L_i$
  - $B_i \rightarrow X \leq U_i$
- Translate $X :: D$ to $B_1 + B_2 + \ldots + B_m = 1$

$X \neq Y$ and abs(X) = Y

- **$X \neq Y$**
  - $B \rightarrow X > Y$
  - $\sim B \rightarrow X < Y$
- abs(X) = Y
  - T = -X
  - Y = max(X, T)

$X \times Y = Z$, X *div* $Y = Z$, X *mod* $Y = Z$

- Let X's binary representation be $\langle X_{n-1}, \ldots, X_1, X_0 \rangle$
- Translate $X \times Y = Z$ to:
  - $Z = 2^{n-1} T_{n-1} + \ldots + 2T_1 + T_0$ Binary expansion
    - $X_i \rightarrow T_i = Y$
    - $\neg X_i \rightarrow T_i = 0$
- Convert $X \ \mathit{div} \ Y = Z$ $(X \geq 0, Y > 0)$ to
  - $X = Y \times Z + R$
    - $0 \leq R < Y$
- Convert $X \ \mathit{mod} \ Y = Z$ $(X \geq 0, Y > 0)$ to
  - $X = Y \times Q + Z$
    - $0 \leq Z < Y$

min(X,Y) = Z

- **min(X,Y) = Z**
  - Z $\leq X$
  - Z $\leq Y$
  - B1 → X ≤ Z
  - B2 → Y ≤ Z
  - B1+B2 $\geq Z$

$B \rightarrow X \leq Y$, $B \leftrightarrow X \leq Y$

- $B \rightarrow X \leq Y$
  - $X - (1-B)M \leq Y$

The big-M method

- $B \leftrightarrow X \leq Y$
  - $B \rightarrow X \leq Y$
  - $\sim B \rightarrow X > Y$

table_in({X1,X2,...,Xn}, T)

$$T = [\{t_{11}, t_{12}, ..., t_{1n}\},\\ \{t_{21}, t_{22}, ..., t_{2n}\},\\ ...\\ \{t_{m1}, t_{m2}, ..., t_{mn}\}]$$

- Introduce a binary variable B_i for each row
  - for $j \in 1..n : X_j = \sum_{i=1}^{m} B_i \times t_{ij}$
  - $\sum_{i=1}^{m} B_i = 1$

https://www.minizinc.org/

Summary

- As demonstrated by Picat, encoding CSP into SAT and MIP are viable and economical ways to having efficient CSP solvers
- The **sat** encoder incorporates cutting-edge encodings and optimizations for constraints
- The **mip** encoder implements some of the standard linearization algorithms
- A logic language, like Picat, is ideal for implementing these encoders

Thank you!

picat-lang.org