

# **SAT: Z3 Exercises**

**Constraint Embeddings**

**Angelo Quarta**

**27 March 2025**

# Table of Contents

1. Cardinality Constraints

2. At Most One

3. At Most K



# Cardinality Constraints

# Cardinality Constraints

In most of the exercises we solved we found a cardinality constraint.

Cardinality constraints are the ones in the form:

$$x_1 + \dots + x_n \leq k$$

# Cardinality Constraints

In sat these constraints are represented by:

- `at_least_one`, `at_most_one` and `exactly_one`;
- `at_least_k`, `at_most_k` and `exactly_k`.

The encodings we presented can be very inefficient with big instances, so we need to find better ones.

# Cardinality Constraints

Keeping in mind that:

- $\text{at\_least\_k}([x_1, \dots, x_n], k) \equiv \text{at\_most\_k}(\{\neg x_i \mid x_i \in [x_1, \dots, x_n]\}, n - k)$
- $\text{exactly\_k}([x_1, \dots, x_n]) \equiv \text{at\_most\_k}([x_1, \dots, x_n]) \wedge \text{at\_least\_k}([x_1, \dots, x_n])$

We are going to focus just on the `at_most_k` constraint.



**At Most One**

# Pairwise Encoding

The pairwise(or naive) encoding of the at\_most\_one constraint is:

$$\bigwedge_{1 \leq i < n} \bigwedge_{i+1 \leq j \leq n} \neg(x_i \wedge x_j)$$

This encoding doesn't require the addition of any new variables, but it encodes  $O(n^2)$  clauses.



# Sequential Encoding

The sequential encoding of the `at_most_one` constraint consists of using  $n - 1$  variables  $s_i$  to keep track of which  $x_i$  is *true*, it is encoded as follows:

$$(\neg x_1 \vee s_1) \wedge (\neg x_n \vee \neg s_{n-1}) \wedge \bigwedge_{1 < i < n} ((\neg x_i \vee s_i) \wedge (\neg s_{i-1} \vee s_i) \wedge (\neg x_i \vee \neg s_{i-1}))$$

This encoding produces  $O(n)$  clauses.

# Bitwise Encoding

The bitwise encoding of the at\_most\_one constraint consists of using  $m = \log_2(n)$  new variables  $r_1, \dots, r_m$  to represent the binary encoding of  $i - 1$ , so it is encoded like:

$$\bigwedge_{1 \leq i \leq n} \bigwedge_{1 \leq j \leq m} \neg x_i \vee r_{i,j} [\neg r_{i,j}]$$

Where  $r_{i,j} [\neg r_{i,j}]$  if bit  $j$  of the binary encoding of  $i - 1$  is 1[0].

This encoding produces  $n \log_2(n)$  clauses.

# Heule Encoding

The Heule encoding is another linear version of the at\_most\_one consisting of splitting the encoding into two parts:

When  $n > 4$  we add an auxiliary variable  $y$  such that:

$$\text{at\_most\_one}(x_1, \dots, x_3, y) \wedge$$

$$\text{at\_most\_one}(\neg y, x_4, \dots, x_n)$$

When  $n \leq 4$ :

$$\bigwedge_{1 \leq i < j \leq n} \neg(x_i \wedge x_j)$$

This encoding require the addition of  $\frac{(n-3)}{2}$  new variable, but it encodes  $3n - 6$  clauses.



**At Most  $K$**

# Pairwise Encoding

The pairwise(or naive) encoding of the at\_most\_k constraint is:

$$\text{at\_most\_k}([x_1, \dots, x_n], k) \equiv \bigwedge_{M \subseteq \{1, \dots, n\}} \bigvee_{i \in M} \neg x_i$$

This encoding doesn't require the addition of any new variables, but it encodes  $\binom{k}{n-1}$  clauses of length  $k + 1$ , with  $|M| = k + 1$ .

# Sequential Encoding

The sequential encoding of the `at_most_k` constraint consists of using  $(n - 1) \times k$  variables  $s_{i,j}$  to keep track of which  $x_i$  is *true*, and what sum  $j$  is reached at index  $i$ . it is encoded as follows:

$$\left. \begin{array}{l} (\neg x_1 \vee s_{1,1}) \\ (\neg s_{1,j}) \quad \text{for } 1 < j \leq k \\ (\neg x_i \vee s_{i,1}) \\ (\neg s_{i-1,1} \vee s_{i,1}) \\ (\neg x_i \vee \neg s_{i-1,j-1} \vee s_{i,j}) \\ (\neg s_{i-1,j} \vee s_{i,j}) \\ (\neg x_i \vee \neg s_{i-1,k}) \\ (\neg x_n \vee \neg s_{n-1,k}) \end{array} \right\} \text{for } 1 < j \leq k \quad \left. \vphantom{\begin{array}{l} (\neg x_1 \vee s_{1,1}) \\ (\neg s_{1,j}) \quad \text{for } 1 < j \leq k \\ (\neg x_i \vee s_{i,1}) \\ (\neg s_{i-1,1} \vee s_{i,1}) \\ (\neg x_i \vee \neg s_{i-1,j-1} \vee s_{i,j}) \\ (\neg s_{i-1,j} \vee s_{i,j}) \\ (\neg x_i \vee \neg s_{i-1,k}) \\ (\neg x_n \vee \neg s_{n-1,k}) \end{array}} \right\} \text{for } 1 < i < n$$

This encoding needs  $2nk + n - 3k - 1$  clauses.