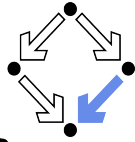


RISC

RESEARCH INSTITUTE FOR
SYMBOLIC COMPUTATION



JKU

JOHANNES KEPLER
UNIVERSITY LINZ

An Intermediate Representation Format for Industrial Optimization Problems - The Translation of OptDSL to MiniZinc

Tereso del Río, Wolfgang Schreiner, Martina
Seidl, Temur Kutsia, Wolfgang Windsteiger

April 2026

RISC Report Series No. 26-04

ISSN: 2791-4267 (online)

Available at <https://doi.org/10.35011/risc.26-04>



This work is licensed under a CC BY 4.0 license.

Editors: RISC Faculty

B. Buchberger, R. Hemmecke, T. Kutsia, G. Landsmann, P. Paule,
V. Pillwein, N. Popov, S. Radu, J. Schicho, C. Schneider, W. Schreiner,
W. Windsteiger, F. Winkler.

Supported by: Supported by the FFG project FO999923579 “InProSSA:
Industrial Problem Solving Using Symbolic and Subsymbolic AI”

**JOHANNES KEPLER
UNIVERSITY LINZ**
Altenberger Str. 69
4040 Linz, Austria
www.jku.at
DVR 0093696

An Intermediate Representation Format for Industrial Optimization Problems*

The Translation of OptDSL to MiniZinc

Tereso del Río^{1,2} Wolfgang Schreiner¹

Martina Seidl²

Temur Kutsia¹ Wolfgang Windsteiger¹

¹ Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria

² Symbolic Artificial Intelligence Institute (SAI)
Johannes Kepler University, Linz, Austria

(Tereso.del.Rio | Wolfgang.Schreiner)@risc.jku.at
martina.seidl@jku.at

(Temur.Kutsia | Wolfgang.Windsteiger)@risc.jku.at

April 1, 2026

Abstract

This report presents the implementation of `OptDSL`, a Python-inspired domain-specific language for describing optimisation problems. The implementation is based on the translation of a high-level `OptDSL` formulation of the problem to an intermediate representation in the constraint modelling language `MiniZinc`, which can be used by multiple state-of-the-art solvers. The report also describes the translation software, illustrates its use on a simplified industrial example, discusses selected implementation details, and suggests directions for further development.

*Supported by the FFG project FO999923579 “InProSSA: Industrial Problem Solving Using Symbolic and Subsymbolic AI”

Contents

1	Introduction	3
2	The Principles of the Translation	3
2.1	Objective	3
2.2	Types	4
2.3	Constants	5
2.4	Static Single Assignment	5
2.5	Constraints	5
2.6	If-else Blocks	6
2.7	For Loops	7
2.8	Logical Expressions	7
2.9	Functions	7
2.10	Function Calls	8
3	Using the Software	10
3.1	Installation	11
3.2	User Functions	11
3.3	Simplified Industrial Example	12
4	The Translation Software	13
4.1	Abstract Syntax Tree	14
4.2	Implementation Details for Conditional Blocks	14
4.3	Implementation Details for Static Single Assignment Forms	15
4.4	Implementation details for Function Calls	15
5	Further Work	16
A	Simplified Industrial Problem	17
B	Full Industrial Problem	51

1 Introduction

In this report, we document the implementation of `OptDSL`, a domain-specific language for formulating optimisation problems in a way that is convenient for industrial users. The language was developed in a joint project with an industrial partner [4] to ease the formulation of optimisation problems by moving away from the declarative style used in most formulation languages. The present report complements the description of the language itself given in [3] and the corresponding conference submission [2] by focusing on its implementation and, in particular, on its translation to the constraint modelling language MiniZinc [5].

The source language is designed as a restricted subset of Python and includes the main constructs needed to describe optimisation problems in a readable and modular way. The target language is MiniZinc, which allows the translated models to be solved by a range of existing backend solvers. The goal of the implementation is to use MiniZinc to bridge the high-level modelling language `OptDSL` and the low-level languages used by standard solvers.

The rest of this report is organised as follows. Section 2 presents the main principles underlying the translation from `OptDSL` into MiniZinc. Section 3 explains how to install and use the software while illustrating its use on a simplified industrial example. Section 4 describes the structure of the translation software and discusses some implementation details. Section 5 outlines directions for further work. The appendices contain formulations of a simplified and a full industrial problem specification.

The implementation and the examples discussed in this report are archived in the Zenodo release at <https://zenodo.org/records/19348397>.

2 The Principles of the Translation

The source language is a subset of Python that includes the main constructs needed to describe optimisation problems in a way that is convenient for industrial users: types, constants, functions, if-else clauses, for loops, and logical operators such as `and`, `or`, and `any`. A detailed description of the grammar of this language can be found in [3].

The target language is MiniZinc. The translation produces MiniZinc type declarations, constants, predicates, variable arrays in static single assignment (SSA) form, and constraints.

The translation runs in two steps. In the first step, constants, types, and functions/predicates are collected. In the second step, variables are transformed into static single assignment form and the corresponding constraints are generated.

2.1 Objective

Any optimisation problem must have an objective function. In the source language, this is expressed using `minimise (...)` or `maximise (...)`. In the generated MiniZinc model, this becomes `solve minimize ...` or `solve maximize ...`.

For example, the following `OptDSL` fragment defines a variable and minimises it.

Listing 1: OptDSL objective

```
a : int
minimize(a)
```

Listing 2: Generated MiniZinc

```
array[1..1] of var int: a;
solve minimize a[1];
```

In case no objective is given, the translation generates a satisfaction problem, that is, it asks MiniZinc to find any solution satisfying all constraints.

Listing 3: OptDSL satisfaction problem

```
a : int
```

Listing 4: Generated MiniZinc

```
array[1..1] of var int: a;
solve satisfy;
```

2.2 Types

The available types are Boolean (DSBool), integer (DSInt), float (DSFloat), list (DSLlist(length, element_type)), and record (DSRecord({"name": type, ...})).

Basic scalar types are translated directly into MiniZinc types.

Listing 5: OptDSL type declarations

```
MyInt = DSInt(3, 7)

MyFloat = DSFloat(lb=0.0,
                  ub=1.0)

Flag = DSBool()
```

Listing 6: Generated MiniZinc

```
type MyInt = 3..7;
type MyFloat = 0.0..1.0;
type Flag = bool;
solve satisfy;
```

Lists are translated into MiniZinc arrays. The default element type is int, but custom element types are also supported.

Listing 7: OptDSL list types

```
Vec = DSLlist(5)

MyInt = DSInt(3, 7)
Vec3 = DSLlist(length=10,
               elem_type=MyInt)
```

Listing 8: Generated MiniZinc

```
type Vec = array[1..5] of int;
type MyInt = 3..7;
type Vec3 = array[1..10] of MyInt;
solve satisfy;
```

Records are translated into MiniZinc record types, and fields may themselves contain lists or user-defined types.

Listing 9: OptDSL record types

```

Prob = DSFloat(lb=0.0, ub
  =1.0)
Vec10 = DSLList(length=10,
  elem_type=int)
Sample = DSRecord({
  "id": "int",
  "values": "Vec10",
  "prob": "Prob"
})

```

Listing 10: Generated MiniZinc

```

type Prob = 0.0..1.0;
type Vec10 = array[1..10] of int;
type Sample = record(
  int: id,
  Vec10: values,
  Prob: prob
);
solve satisfy;

```

2.3 Constants

Constants must be given in all capital letters and are translated directly into MiniZinc constant declarations. They may use built-in types or previously defined custom types.

Listing 11: OptDSL constants

```

MAX_N : int = 10
LB : int = 0
UB : int = MAX_N
ValueType = DSLList(2, int)
VALUES: ValueType = [1, 2]
.

```

Listing 12: Generated MiniZinc

```

int: MAX_N = 10;
int: LB = 0;
int: UB = 10;
type ValueType = array[1..2] of int;
ValueType: VALUES = [1, 2];
solve satisfy;

```

2.4 Static Single Assignment

We use static single assignment form, meaning that every time a variable is assigned or reassigned, a new version of that variable is created. Note that variables must be given a type definition as well. For example:

Listing 13: OptDSL reassignment

```

x : int = 0
x = x + 1
.

```

Listing 14: Generated MiniZinc

```

array[1..2] of var int: x;
constraint x[1] = 0;
constraint x[2] = (x[1] + 1);
solve satisfy;

```

Note that the target language uses `constraint` statements. Assignments in the source language therefore become equality constraints in MiniZinc.

2.5 Constraints

Assertions and assignments generate constraints in the target language. Assertions generate constraints without creating a new version, while assignments create equality constraints in a new version of the variable.

For example, an assertion such as `assert x > 0` becomes a MiniZinc constraint over the current version of `x`.

Listing 15: OptDSL assertion

```
x : int = 1
assert x > 0
.
```

Listing 16: Generated MiniZinc

```
array[1..1] of var int: x;
constraint x[1] = 1;
constraint (x[1] > 0);
solve satisfy;
```

`if` or `else` blocks add conditions to the generated constraints.

2.6 If-else Blocks

An `if` without an `else` generates one constraint for the true branch and one propagation constraint for the false branch.

Listing 17: OptDSL if without else

```
x = 0
if x > 0:
    x = x + 1
.
```

Listing 18: Generated MiniZinc

```
array[1..2] of var int: x;
constraint x[1] = 0;
constraint (x[1] > 0)
    -> x[2] = (x[1] + 1);
constraint (not (x[1] > 0))
    -> x[2] = x[1];
solve satisfy;
```

An `if-else` block generates one guarded constraint for each branch. Variable types must remain consistent across branches.

Listing 19: OptDSL if-else

```
x = 0
if x > 0:
    x = x + 1
else:
    x = x - 1
    x = x - 2
    x = x - 3
.
```

Listing 20: Generated MiniZinc

```
array[1..2] of var int: x;
constraint x[1] = 0;
constraint (x[1] > 0)
    -> x[2] = (x[1] + 1);
constraint (x[1] > 0) -> x[4] = x[2];
constraint (not (x[1] > 0))
    -> x[2] = (x[1] - 1);
constraint (not (x[1] > 0))
    -> x[3] = (x[2] - 2);
constraint (not (x[1] > 0))
    -> x[4] = (x[3] - 3);
solve satisfy;
```

Not that even though the branches do not create the same number of versions, the version is uniformised to `x[4]`.

2.7 For Loops

A for loop with known bounds is unfolded into multiple constraints, one for each iteration. This matches the way the user writes the model, while producing explicit MiniZinc constraints.

For example, a loop over a range becomes a sequence of assignments over successive SSA versions.

Listing 21: OptDSL range loop

```
x = 0
for i in range(1, 4):
    x = x + i
.
```

Listing 22: Generated MiniZinc

```
array[1..4] of var int: x;
constraint x[1] = 0;
constraint x[2] = (x[1] + 1);
constraint x[3] = (x[2] + 2);
constraint x[4] = (x[3] + 3);
solve satisfy;
```

Loops over lists are also unfolded element by element.

Listing 23: OptDSL loop over list

```
x = 0
for t in [3, 1, 5]:
    x = x + t
.
```

Listing 24: Generated MiniZinc

```
array[1..4] of var int: x;
constraint x[1] = 0;
constraint x[2] = (x[1] + 3);
constraint x[3] = (x[2] + 1);
constraint x[4] = (x[3] + 5);
solve satisfy;
```

2.8 Logical Expressions

Logical expressions such as any are translated into an expanded disjunction.

Listing 25: OptDSL any assertion

```
x : DSList(5, int)
assert any(x[cut] > 0
    for cut in range(1, 6))
.
```

Listing 26: Generated MiniZinc

```
array[1..1] of array[1..5] of var int: x
;
constraint ((x[1][1] > 0) \\/
    (x[1][2] > 0) \\/
    (x[1][3] > 0) \\/
    (x[1][4] > 0) \\/
    (x[1][5] > 0));
solve satisfy;
```

2.9 Functions

Functions are translated into MiniZinc predicates. The returned value is represented as an output argument of the predicate. Since MiniZinc predicates do not create local decision variables

Listing 29: OptDSL function call

```

def f(a, b):

    c = a + b
    return c

c = f(1, 2)
.

```

Listing 30: Generated MiniZinc

```

predicate f(var int: input_1,
            var int: input_2,
            var int: output_1,
            array[1..1] of var int: a,
            array[1..1] of var int: b,
            array[1..1] of var int: c)
    =
    (
    a[1] = input_1 /\
    b[1] = input_2 /\
    c[1] = (a[1] + b[1]) /\
    output_1 = c[1]
    );
array[1..1] of var int: c;
array[1..1] of var int: a__f__1;
array[1..1] of var int: b__f__1;
array[1..1] of var int: c__f__1;
constraint f(1, 2, c[1], a__f__1,
            b__f__1, c__f__1);
solve satisfy;

```

When the same function is called several times, new auxiliary variables are created for each call.

Listing 31: OptDSL repeated calls

```

def f(a, b):

    c = a + b
    d = a * b
    c = c * d
    return c, d

c : int
d : int

e : int
g : int

c, d = f(1, 2)

e, g = f(c, d)

assert c > d
.

```

Listing 32: Generated MiniZinc

```

predicate f(var int: input_1,
            var int: input_2,
            var int: output_1,
            var int: output_2,
            array[1..1] of var int: a,
            array[1..1] of var int: b,
            array[1..2] of var int: c,
            array[1..1] of var int: d)
    =
    (
    a[1] = input_1 /\
    b[1] = input_2 /\
    c[1] = (a[1] + b[1]) /\
    d[1] = (a[1] * b[1]) /\
    c[2] = (c[1] * d[1]) /\
    output_1 = c[2] /\
    output_2 = d[1]
    );
array[1..1] of var int: c;
array[1..1] of var int: d;
array[1..1] of var int: a__f__1;
array[1..1] of var int: b__f__1;
array[1..2] of var int: c__f__1;
array[1..1] of var int: d__f__1;
array[1..1] of var int: e;
array[1..1] of var int: g;
array[1..1] of var int: a__f__2;
array[1..1] of var int: b__f__2;
array[1..2] of var int: c__f__2;
array[1..1] of var int: d__f__2;
constraint f(1, 2, c[1], d[1],
             a__f__1, b__f__1,
             c__f__1, d__f__1);
constraint f(c[1], d[1], e[1], g[1],
             a__f__2, b__f__2,
             c__f__2, d__f__2);
constraint (c[1] > d[1]);
solve satisfy;

```

3 Using the Software

In this section, we describe how to install the software, the general functions available to the user, and a simplified industrial example.

```

851 array[1..11] of array[1..10] of var 0..MAX_BOARD_LENGTH: cut_pieces_cutting_stage_1;
852 array[1..1] of array[1..5] of var 0..MAX_BOARD_LENGTH: cut_positions_cutting_stage_1;
853 array[1..6] of var 0..N_BOARDS: cutting_cost_cutting_stage_1;
854 array[1..1] of array[1..10] of var 1..MAX_N_LAYERS: Assignments;
855 array[1..2] of var 0..(N_BOARDS * 6): use_cost;
856 array[1..1] of array[1..10] of var 1..MAX_N_LAYERS: assignments_packing_stage_1;
857 array[1..1] of array[1..10] of var 0..MAX_BOARD_LENGTH: cut_pieces_packing_stage_1;
858 array[1..11] of var 0..(N_BOARDS * 6): use_cost_packing_stage_1;
859 array[1..10] of array[1..10] of var 0..sum(ORIGINAL_BOARDS): used_length_packing_stage_1;
860 array[1..1] of var 0..(N_BOARDS * 7): total_cost;
861 constraint cutting_stage(CutPositions[1], cut_pieces[1], cutting_cost[1], cut_pieces_cutting_stage_1, cut_positions_cutting_stage_1, cutting_
862 constraint use_cost[1] = 0;
863 constraint packing_stage(Assignments[1], cut_pieces[1], use_cost[2], assignments_packing_stage_1, cut_pieces_packing_stage_1, use_cost_packi
864 constraint total_cost[1] = (cutting_cost[1] + use_cost[2]);
865 solve minimize total_cost[1];

```

total_cost = [14];

Figure 1: Screenshot of MiniZinc Playground

3.1 Installation

The implementation and the examples discussed in this report are archived in the Zenodo release at <https://zenodo.org/records/19348397>. To run the software, the user downloads and unpacks the archive, opens a terminal in the project directory, and installs the Python dependencies listed in `requirements.txt`. The software can then be run from the command line as follows.

Listing 33: Running the software on Linux and Windows

```

Linux:
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
python3 main.py

Windows:
py -m venv .venv
.venv\Scripts\activate
pip install -r requirements.txt
py main.py

```

3.2 User Functions

Two objects are designed for the user to use: `MiniZincTranslator` and `DSLsSolver`. The first one takes a string of `OPTDSL` code and has the function `unroll_translation()` that generates the translated MiniZinc code. The second one also takes the code as input, internally translates it, and has a function `run()` that runs the translated MiniZinc code and returns the result.

Users can easily use the software by replacing the code in `main.py` with their own. The current code contains a simplified industrial example that is covered in the following. Another option for the users is to translate their code using `MiniZincTranslator.unroll_translation()` and input it into [MiniZinc Playground](#) as shown in [Figure 1](#)

3.3 Simplified Industrial Example

RISC Software [6], the industrial partner in this project, proposed a substantially more complex optimisation problem, whose full formulation and translation are presented in Appendix B. In this section, we consider a simplified version that captures the main structure of the original task. The simplified setting is illustrated in Figure 2.

In this problem, a set of one-dimensional boards is given. Each board may either remain uncut or be cut once into two pieces. The resulting pieces must then be assigned to layers of fixed length. Each cut incurs a cost of 1, and each layer that is used incurs a cost of 3. The goal is to determine the cuts and assignments that minimise the total cost.

A complete `OPrDSL` specification of this simplified problem is provided in Appendix B. In the present section, we only show the most relevant fragments, omitting some type annotations for readability.

We begin by defining the constants that describe the instance shown in Figure 2.

```
LAYER_LENGTH : DSInt() = 17
N_BOARDS    : int    = 5
ORIGINAL_BOARDS : DSList(N_BOARDS, DSInt()) = [12, 15, 13, 10, 15]
MAX_BOARD_LENGTH : int    = 15
MAX_N_LAYERS  : int    = N_BOARDS * 2
```

The decision variables are then introduced. Following the convention used in this example, their names are written in camel case to distinguish them from auxiliary variables and constants.

```
CutPositions: DSList(N_BOARDS, DSInt(0, MAX_BOARD_LENGTH))
Assignments: DSList(N_BOARDS * 2, DSInt(1, MAX_N_LAYERS))
```

One of the advantages of `OPrDSL` is that it allows the two-stage structure of the problem to be expressed directly and modularly.

```
cut_pieces, cutting_cost = cutting_stage(CutPositions)
use_cost = packing_stage(Assignments, cut_pieces)
minimize(cutting_cost + use_cost)
```

The function `cutting_stage` takes the decision variable `CutPositions`, while the board data are provided as global constants. It returns both the generated pieces and the cutting cost. The function `packing_stage` then takes these pieces together with the decision variable `Assignments`, which determines to which layer each piece is assigned, and returns the packing cost based on the number of layers used. The optimisation problem therefore consists of finding values for `CutPositions` and `Assignments` that minimise the sum of both costs.

The cutting stage is defined as follows.

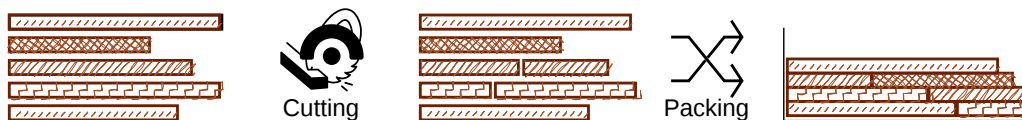


Figure 2: Illustrative instance of the simplified cutting-and-packing problem.

```

def cutting_stage(cut_positions):
    cutting_cost = 0
    for n_board in range(1, N_BOARDS + 1):
        assert cut_positions[n_board] <= ORIGINAL_BOARDS[n_board]
        if cut_positions[n_board] != 0:
            cutting_cost = cutting_cost + 1
        cut_pieces[2 * n_board] = ORIGINAL_BOARDS[n_board] -
            cut_positions[n_board]
        cut_pieces[2 * n_board - 1] = cut_positions[n_board]
    return cut_pieces, cutting_cost

```

In this stage, the model iterates over all boards, ensuring through an assertion that the cut position does not exceed the board length. Whenever a nonzero cut is made, the cutting cost is increased by 1. Each board then gives rise to two pieces; if no cut is performed, one of these pieces has length 0.

The packing stage is specified as follows.

```

def packing_stage(assignments, cut_pieces):
    use_cost = 0
    for n_layer in range(1, MAX_N_LAYERS + 1):
        used_length[n_layer] = 0
        for n_board in range(1, N_BOARDS * 2 + 1):
            if assignments[n_board] == n_layer:
                used_length[n_layer] = used_length[n_layer] +
                    cut_pieces[n_board]
        assert used_length[n_layer] <= LAYER_LENGTH
        if used_length[n_layer] > 0:
            use_cost = use_cost + 3
    return use_cost

```

In this second stage, the model iterates over the possible layers and accumulates the total length of all pieces assigned to each layer. An assertion enforces that the occupied length of a layer does not exceed the allowed layer length. Whenever a layer contains at least one piece, a cost of 3 is added.

This example illustrates how OPTDSL can represent the behaviour of processing stages while embedding feasibility conditions in a natural and readable way. Rather than requiring the modeller to derive a purely declarative formulation from the outset, the language allows the problem to be expressed in a checker-like style that is often closer to how practitioners describe industrial processes. At the same time, the use of functions supports modularity, and the Python-like syntax makes the formulation easier to read, debug, and communicate. Although strong type annotations are still required, OPTDSL formulations retain all the information needed for translation into a solver-ready model.

4 The Translation Software

This section describes the organisation of the translation software, its implementation, and its use in practice.

At the highest level, the translation process is managed by the `MiniZincTranslator` class. This class is responsible for traversing the top level of the Python abstract syntax tree and separating the input program into its main components: constants, type definitions, function or predicate definitions, the objective specification, and the remaining executable statements.

The parsing stage is carried out by `self.parse()`. In this stage, constant definitions are identified from annotated assignments with uppercase names, their values are processed through `CodeBlock`, and the resulting objects are stored in `self.constants`. Type definitions are processed through `DSType` and stored in `self.types`. Function definitions are processed through `Predicate` and stored in `self.predicates`. If an optimisation objective is present, it is stored in `self.objective`. All remaining top-level statements are collected in `self.top_level_stmts` for subsequent translation.

After parsing, the method `self.compile()` generates the final MiniZinc model. Using the constants, types, and predicates collected in the first stage, it compiles the remaining top-level statements and produces a single MiniZinc file containing type declarations, predicate definitions, constants, decision variables, constraints, and the objective.

The class `CodeBlock` is responsible for translating a list of statements within a given context, that is, using the constants, types, and functions already known at that point. Its main entry point is `run()`, which processes a block of abstract syntax tree statements and collects the constants, variables, and constraints generated inside that block. Internally, the recursive method `execute_block()` handles the supported statement forms, including assignments, annotated assignments, for loops, `if` statements, assertions, and function calls.

4.1 Abstract Syntax Tree

Because the source language is designed as a restricted subset of Python, the implementation can make direct use of Python's abstract syntax tree. This provides a convenient and robust intermediate representation of the source program. Rather than building a custom parser from scratch, the translator relies on Python's `ast` module to parse the input and expose its structural components. This makes it possible to process assignments, control flow, function definitions, and expressions in a systematic way while preserving a syntax that is familiar to the user.

4.2 Implementation Details for Conditional Blocks

Conditional blocks require special care because different branches may create or update different variables. At the end of an `if-else` block, the method responsible for conditional execution collects the variables produced in each branch and merges them into a consistent post-branch state. During this merge, the translator verifies that variables with the same name have compatible types in both branches. In the current implementation, matching types are required. If the type system is extended in the future, this merge procedure should be adapted accordingly so that the resulting merged type correctly reflects the information coming from both branches.

4.3 Implementation Details for Static Single Assignment Forms

The static single assignment transformation must be handled carefully for structured objects such as lists and records. In particular, when only a subset of the components of a list or record is assigned for the first time, no new version is created. A new version is created only when a previously assigned component is modified again. In that case, the updated version contains the new assigned value together with equality constraints that copy over the unchanged components from the previous version.

Listing 34: OptDSL function

```
a : DSLList(3)
a[1] = 7
a[2] = 9
a[1] = 11

a[3] = 17
```

Listing 35: Generated MiniZinc

```
array[1..2] of array[1..3] of var int: d;
a[1][1] = 7
a[1][2] = 9
a[2][1] = 11
a[2][2] = a[1][2]
a[2][3] = 17
```

This optimisation is optional, but it is useful in practice because it significantly reduces the number of generated SSA versions and therefore keeps the translated MiniZinc model more compact, otherwise the translation of the previous code would look as follows:

Listing 36: OptDSL function

```
a : DSLList(3)
a[1] = 7
a[2] = 9

a[1] = 11

a[3] = 17
```

Listing 37: Generated MiniZinc

```
array[1..4] of array[1..3] of var int: d;
a[1][1] = 7
a[2][2] = 9
a[2][1] = a[1][1]
a[2][3] = a[1][3]
a[3][1] = 11
a[3][2] = a[2][2]
a[3][3] = a[2][3]
a[4][3] = 17
a[4][1] = a[3][1]
a[4][2] = a[3][2]
```

4.4 Implementation details for Function Calls

Function calls are translated into predicate calls. To ensure that generated MiniZinc code remains unambiguous, both inputs and outputs are assigned explicit internal names. In addition, auxiliary variables representing the internal state of each function call are renamed systematically using the function name and the call number. This naming scheme avoids collisions when the same function is called multiple times within the same model and makes the generated code easier to trace during debugging.

5 Further Work

Several directions remain for extending OPTDSL.

First, the language could be made executable in Python. In particular, 0-based indexing could be adapted and current custom type declarations could be replaced with Python-compatible annotations. This would allow OPTDSL code to serve as a checker for candidate solutions.

Type inference could replace explicit type annotations in some cases, making the language more concise and accessible without changing its expressive power.

Decision variables could be made more explicit by enforcing a dedicated naming convention, such as camel case, and their values could be automatically extracted from solver outputs. OPTDSL would then be easier to use, and no manual post-processing would be required from users.

There is the question of automatically generating stronger formulations. Large language models could be used not only to assist in writing OPTDSL programs, but also to search for alternative formulations that lead to faster solving times or better solver performance [1, 2]. Since the formulation follows a checker-style, any solution provided by an automatically generated formulation could be validated using the original problem formulation.

Finally, translation could be extended to other optimisation backends, such as Gurobi or OR-Tools. But more importantly, this formulation could be used to automatically derive heuristics, rather than MILP formulations. This is because in many industrial applications tailored heuristic approaches completely outperform solving optimisation problems using generic search algorithms.

References

- [1] Katharina Blaimschein. Autoformulation of Optimization Problems with Trees of Thought for LLMs. Master’s thesis, Johannes Kepler University Linz, Linz, Austria, 2026.
- [2] Tereso del Río, Katharina Blaimschein, Martina Seidl, and Wolfgang Schreiner. The Pythonic Way: Automatising the Formulation of Industrial Optimisation Problems with LLMs. Paper under review for the International Conference on Principles and Practice of Constraint Programming (CP 2026), 2026.
- [3] Tereso del Rio, Wolfgang Schreiner, Martina Seidl, Temur Kutsia, and Wolfgang Windsteiger. A DSL for Specifying a Class of Industrial Optimisation Problems. Technical Report 25-12, Research Institute for Symbolic Computation (RISC), Johannes Kepler University Linz, Linz, Austria, December 2025.
- [4] InProSSA: Integration of Symbolic and Subsymbolic AI for Industry. RISC Software, RISC Institute, Institute for Symbolic AI, 2025. <https://www.risc-software.at/en/referenceprojects/research-project-inprossa>.
- [5] Nicholas Nethercote, Peter J. Stuckey, Guido Tack, Sebastian Brand, Gregory J. Duck, and Rémy Youssef. MiniZinc: Towards a Standard CP Modelling Language. In *Principles and Practice of Constraint Programming (CP 2007)*, pages 529–543. Springer, 2007.
- [6] RISC Software GmbH. <https://www.risc-software.at/>.

A Simplified Industrial Problem

The full formulation of the wood-cutting problem in [Figure 2](#). To obtain a translation, please use the associated repository.

The OptDSL Source

```
"""
Showcasing how to run an optimisation given an OptDSL formulation
"""

from src import MiniZincTranslator
from src import DSLSolver

dsl_code = f"""
LAYER_LENGTH : DSInt() = 17
N_BOARDS : int = 5
ORIGINAL_BOARDS : DSLList(N_BOARDS, DSInt()) = [12, 15, 13, 10, 15]
MAX_BOARD_LENGTH : int = 15
MAX_N_LAYERS : int = N_BOARDS * 2

def cutting_stage(
    cut_positions: DSLList(N_BOARDS, DSInt(0, MAX_BOARD_LENGTH))
):
    cutting_cost: DSInt(0, N_BOARDS) = 0
    cut_pieces: DSLList(N_BOARDS * 2, DSInt(0, MAX_BOARD_LENGTH))
    for n_board in range(1, N_BOARDS + 1):
        assert cut_positions[n_board] <= ORIGINAL_BOARDS[n_board]
        cut_pieces[2 * n_board] = ORIGINAL_BOARDS[n_board] -
            cut_positions[n_board]
        cut_pieces[2 * n_board - 1] = cut_positions[n_board]
        if cut_positions[n_board] != 0:
            cutting_cost = cutting_cost + 1
    return cut_pieces, cutting_cost

def packing_stage(
    assignments: DSLList(N_BOARDS * 2, DSInt(1, MAX_N_LAYERS)),
    cut_pieces: DSLList(N_BOARDS * 2, DSInt(0, MAX_BOARD_LENGTH))
):
    use_cost: DSInt(0, N_BOARDS * 6) = 0
    used_length: DSLList(MAX_N_LAYERS, DSInt(0, sum(ORIGINAL_BOARDS)))
    for n_layer in range(1, MAX_N_LAYERS + 1):
        used_length[n_layer] = 0
        for n_board in range(1, N_BOARDS * 2 + 1):
            if assignments[n_board] == n_layer:
                used_length[n_layer] = used_length[n_layer] +
                    cut_pieces[n_board]
        assert used_length[n_layer] <= LAYER_LENGTH

```

```

        if used_length[n_layer] > 0:
            use_cost = use_cost + 3
    return use_cost

CutPositions: DSLList(N_BOARDS, DSInt(0, MAX_BOARD_LENGTH))
cut_pieces: DSLList(N_BOARDS * 2, DSInt(0, MAX_BOARD_LENGTH))
cutting_cost : DSInt(0, N_BOARDS)
cut_pieces, cutting_cost = cutting_stage(CutPositions)

Assignments: DSLList(N_BOARDS * 2, DSInt(1, MAX_N_LAYERS))
use_cost: DSInt(0, N_BOARDS * 6) = 0
use_cost = packing_stage(Assignments, cut_pieces)
total_cost: DSInt(0, N_BOARDS * 7) = cutting_cost + use_cost

minimize(total_cost)
"""

if __name__ == "__main__":
    minizinc_translator = MiniZincTranslator(dsl_code)
    print(minizinc_translator.unroll_translation()) # print the
        translation
    dsl_solver = DSLSolver(dsl_code)
    result = dsl_solver.run()
    print(result["statistics"]["objective"]) # print the optimal
        objective value

```

The generated MiniZinc translation of this OprDSL code using our code is:

Listing 38: Generated MiniZinc

```

% Use this editor as a MiniZinc scratch book
predicate cutting_stage(array[1..5] of var 0..MAX_BOARD_LENGTH:
    input_1, array[1..10] of var 0..MAX_BOARD_LENGTH: output_1, var
    0..N_BOARDS: output_2, array[1..11] of array[1..10] of var 0..
    MAX_BOARD_LENGTH: cut_pieces, array[1..1] of array[1..5] of var
    0..MAX_BOARD_LENGTH: cut_positions, array[1..6] of var 0..N_BOARDS
: cutting_cost) =
(
    cut_positions[1][1] = input_1[1] /\
    cut_positions[1][2] = input_1[2] /\
    cut_positions[1][3] = input_1[3] /\
    cut_positions[1][4] = input_1[4] /\
    cut_positions[1][5] = input_1[5] /\
    cutting_cost[1] = 0 /\
    (cut_positions[1][1] <= ORIGINAL_BOARDS[1]) /\
    cut_pieces[2][(2 * 1)] = (ORIGINAL_BOARDS[1] - cut_positions
        [1][1]) /\
    cut_pieces[3][1] = cut_pieces[2][1] /\
    cut_pieces[3][2] = cut_pieces[2][2] /\
    cut_pieces[3][3] = cut_pieces[2][3] /\

```

```

cut_pieces[3][4] = cut_pieces[2][4] /\
cut_pieces[3][5] = cut_pieces[2][5] /\
cut_pieces[3][6] = cut_pieces[2][6] /\
cut_pieces[3][7] = cut_pieces[2][7] /\
cut_pieces[3][8] = cut_pieces[2][8] /\
cut_pieces[3][9] = cut_pieces[2][9] /\
cut_pieces[3][10] = cut_pieces[2][10] /\
cut_pieces[3][((2 * 1) - 1)] = cut_positions[1][1] /\
((cut_positions[1][1] != 0) -> cutting_cost[2] = (cutting_cost[1]
+ 1)) /\
((not (cut_positions[1][1] != 0)) -> cutting_cost[2] =
cutting_cost[1]) /\
(cut_positions[1][2] <= ORIGINAL_BOARDS[2]) /\
cut_pieces[4][1] = cut_pieces[3][1] /\
cut_pieces[4][2] = cut_pieces[3][2] /\
cut_pieces[4][3] = cut_pieces[3][3] /\
cut_pieces[4][4] = cut_pieces[3][4] /\
cut_pieces[4][5] = cut_pieces[3][5] /\
cut_pieces[4][6] = cut_pieces[3][6] /\
cut_pieces[4][7] = cut_pieces[3][7] /\
cut_pieces[4][8] = cut_pieces[3][8] /\
cut_pieces[4][9] = cut_pieces[3][9] /\
cut_pieces[4][10] = cut_pieces[3][10] /\
cut_pieces[4][(2 * 2)] = (ORIGINAL_BOARDS[2] - cut_positions
[1][2]) /\
cut_pieces[5][1] = cut_pieces[4][1] /\
cut_pieces[5][2] = cut_pieces[4][2] /\
cut_pieces[5][3] = cut_pieces[4][3] /\
cut_pieces[5][4] = cut_pieces[4][4] /\
cut_pieces[5][5] = cut_pieces[4][5] /\
cut_pieces[5][6] = cut_pieces[4][6] /\
cut_pieces[5][7] = cut_pieces[4][7] /\
cut_pieces[5][8] = cut_pieces[4][8] /\
cut_pieces[5][9] = cut_pieces[4][9] /\
cut_pieces[5][10] = cut_pieces[4][10] /\
cut_pieces[5][((2 * 2) - 1)] = cut_positions[1][2] /\
((cut_positions[1][2] != 0) -> cutting_cost[3] = (cutting_cost[2]
+ 1)) /\
((not (cut_positions[1][2] != 0)) -> cutting_cost[3] =
cutting_cost[2]) /\
(cut_positions[1][3] <= ORIGINAL_BOARDS[3]) /\
cut_pieces[6][1] = cut_pieces[5][1] /\
cut_pieces[6][2] = cut_pieces[5][2] /\
cut_pieces[6][3] = cut_pieces[5][3] /\
cut_pieces[6][4] = cut_pieces[5][4] /\
cut_pieces[6][5] = cut_pieces[5][5] /\
cut_pieces[6][6] = cut_pieces[5][6] /\
cut_pieces[6][7] = cut_pieces[5][7] /\
cut_pieces[6][8] = cut_pieces[5][8] /\

```

```

cut_pieces[6][9] = cut_pieces[5][9] /\
cut_pieces[6][10] = cut_pieces[5][10] /\
cut_pieces[6][(2 * 3)] = (ORIGINAL_BOARDS[3] - cut_positions
  [1][3]) /\
cut_pieces[7][1] = cut_pieces[6][1] /\
cut_pieces[7][2] = cut_pieces[6][2] /\
cut_pieces[7][3] = cut_pieces[6][3] /\
cut_pieces[7][4] = cut_pieces[6][4] /\
cut_pieces[7][5] = cut_pieces[6][5] /\
cut_pieces[7][6] = cut_pieces[6][6] /\
cut_pieces[7][7] = cut_pieces[6][7] /\
cut_pieces[7][8] = cut_pieces[6][8] /\
cut_pieces[7][9] = cut_pieces[6][9] /\
cut_pieces[7][10] = cut_pieces[6][10] /\
cut_pieces[7][((2 * 3) - 1)] = cut_positions[1][3] /\
((cut_positions[1][3] != 0) -> cutting_cost[4] = (cutting_cost[3]
  + 1)) /\
((not (cut_positions[1][3] != 0)) -> cutting_cost[4] =
  cutting_cost[3]) /\
(cut_positions[1][4] <= ORIGINAL_BOARDS[4]) /\
cut_pieces[8][1] = cut_pieces[7][1] /\
cut_pieces[8][2] = cut_pieces[7][2] /\
cut_pieces[8][3] = cut_pieces[7][3] /\
cut_pieces[8][4] = cut_pieces[7][4] /\
cut_pieces[8][5] = cut_pieces[7][5] /\
cut_pieces[8][6] = cut_pieces[7][6] /\
cut_pieces[8][7] = cut_pieces[7][7] /\
cut_pieces[8][8] = cut_pieces[7][8] /\
cut_pieces[8][9] = cut_pieces[7][9] /\
cut_pieces[8][10] = cut_pieces[7][10] /\
cut_pieces[8][(2 * 4)] = (ORIGINAL_BOARDS[4] - cut_positions
  [1][4]) /\
cut_pieces[9][1] = cut_pieces[8][1] /\
cut_pieces[9][2] = cut_pieces[8][2] /\
cut_pieces[9][3] = cut_pieces[8][3] /\
cut_pieces[9][4] = cut_pieces[8][4] /\
cut_pieces[9][5] = cut_pieces[8][5] /\
cut_pieces[9][6] = cut_pieces[8][6] /\
cut_pieces[9][7] = cut_pieces[8][7] /\
cut_pieces[9][8] = cut_pieces[8][8] /\
cut_pieces[9][9] = cut_pieces[8][9] /\
cut_pieces[9][10] = cut_pieces[8][10] /\
cut_pieces[9][((2 * 4) - 1)] = cut_positions[1][4] /\
((cut_positions[1][4] != 0) -> cutting_cost[5] = (cutting_cost[4]
  + 1)) /\
((not (cut_positions[1][4] != 0)) -> cutting_cost[5] =
  cutting_cost[4]) /\
(cut_positions[1][5] <= ORIGINAL_BOARDS[5]) /\
cut_pieces[10][1] = cut_pieces[9][1] /\

```

```

cut_pieces[10][2] = cut_pieces[9][2] /\
cut_pieces[10][3] = cut_pieces[9][3] /\
cut_pieces[10][4] = cut_pieces[9][4] /\
cut_pieces[10][5] = cut_pieces[9][5] /\
cut_pieces[10][6] = cut_pieces[9][6] /\
cut_pieces[10][7] = cut_pieces[9][7] /\
cut_pieces[10][8] = cut_pieces[9][8] /\
cut_pieces[10][9] = cut_pieces[9][9] /\
cut_pieces[10][10] = cut_pieces[9][10] /\
cut_pieces[10][(2 * 5)] = (ORIGINAL_BOARDS[5] - cut_positions
  [1][5]) /\
cut_pieces[11][1] = cut_pieces[10][1] /\
cut_pieces[11][2] = cut_pieces[10][2] /\
cut_pieces[11][3] = cut_pieces[10][3] /\
cut_pieces[11][4] = cut_pieces[10][4] /\
cut_pieces[11][5] = cut_pieces[10][5] /\
cut_pieces[11][6] = cut_pieces[10][6] /\
cut_pieces[11][7] = cut_pieces[10][7] /\
cut_pieces[11][8] = cut_pieces[10][8] /\
cut_pieces[11][9] = cut_pieces[10][9] /\
cut_pieces[11][10] = cut_pieces[10][10] /\
cut_pieces[11][((2 * 5) - 1)] = cut_positions[1][5] /\
((cut_positions[1][5] != 0) -> cutting_cost[6] = (cutting_cost[5]
  + 1)) /\
((not (cut_positions[1][5] != 0)) -> cutting_cost[6] =
  cutting_cost[5]) /\
output_1 = cut_pieces[11] /\
output_2 = cutting_cost[6]
);
predicate packing_stage(array[1..10] of var 1..MAX_N_LAYERS: input_1,
  array[1..10] of var 0..MAX_BOARD_LENGTH: input_2, var 0..(
N_BOARDS * 6): output_1, array[1..1] of array[1..10] of var 1..
MAX_N_LAYERS: assignments, array[1..1] of array[1..10] of var 0..
MAX_BOARD_LENGTH: cut_pieces, array[1..11] of var 0..(N_BOARDS *
6): use_cost, array[1..101] of array[1..10] of var 0..sum(
ORIGINAL_BOARDS): used_length) =
(
  assignments[1][1] = input_1[1] /\
  assignments[1][2] = input_1[2] /\
  assignments[1][3] = input_1[3] /\
  assignments[1][4] = input_1[4] /\
  assignments[1][5] = input_1[5] /\
  assignments[1][6] = input_1[6] /\
  assignments[1][7] = input_1[7] /\
  assignments[1][8] = input_1[8] /\
  assignments[1][9] = input_1[9] /\
  assignments[1][10] = input_1[10] /\
  cut_pieces[1][1] = input_2[1] /\
  cut_pieces[1][2] = input_2[2] /\

```

```

cut_pieces[1][3] = input_2[3] /\
cut_pieces[1][4] = input_2[4] /\
cut_pieces[1][5] = input_2[5] /\
cut_pieces[1][6] = input_2[6] /\
cut_pieces[1][7] = input_2[7] /\
cut_pieces[1][8] = input_2[8] /\
cut_pieces[1][9] = input_2[9] /\
cut_pieces[1][10] = input_2[10] /\
use_cost[1] = 0 /\
used_length[1][1] = 0 /\
((assignments[1][1] = 1) -> used_length[2][1] = (used_length
  [1][1] + cut_pieces[1][1])) /\
((not (assignments[1][1] = 1)) -> used_length[2] = used_length
  [1]) /\
((assignments[1][2] = 1) -> used_length[3][1] = (used_length
  [2][1] + cut_pieces[1][2])) /\
((not (assignments[1][2] = 1)) -> used_length[3] = used_length
  [2]) /\
((assignments[1][3] = 1) -> used_length[4][1] = (used_length
  [3][1] + cut_pieces[1][3])) /\
((not (assignments[1][3] = 1)) -> used_length[4] = used_length
  [3]) /\
((assignments[1][4] = 1) -> used_length[5][1] = (used_length
  [4][1] + cut_pieces[1][4])) /\
((not (assignments[1][4] = 1)) -> used_length[5] = used_length
  [4]) /\
((assignments[1][5] = 1) -> used_length[6][1] = (used_length
  [5][1] + cut_pieces[1][5])) /\
((not (assignments[1][5] = 1)) -> used_length[6] = used_length
  [5]) /\
((assignments[1][6] = 1) -> used_length[7][1] = (used_length
  [6][1] + cut_pieces[1][6])) /\
((not (assignments[1][6] = 1)) -> used_length[7] = used_length
  [6]) /\
((assignments[1][7] = 1) -> used_length[8][1] = (used_length
  [7][1] + cut_pieces[1][7])) /\
((not (assignments[1][7] = 1)) -> used_length[8] = used_length
  [7]) /\
((assignments[1][8] = 1) -> used_length[9][1] = (used_length
  [8][1] + cut_pieces[1][8])) /\
((not (assignments[1][8] = 1)) -> used_length[9] = used_length
  [8]) /\
((assignments[1][9] = 1) -> used_length[10][1] = (used_length
  [9][1] + cut_pieces[1][9])) /\
((not (assignments[1][9] = 1)) -> used_length[10] = used_length
  [9]) /\
((assignments[1][10] = 1) -> used_length[11][1] = (used_length
  [10][1] + cut_pieces[1][10])) /\
((not (assignments[1][10] = 1)) -> used_length[11] = used_length

```

```

[10]) /\
(used_length[11][1] <= LAYER_LENGTH) /\
((used_length[11][1] > 0) -> use_cost[2] = (use_cost[1] + 3)) /\
((not (used_length[11][1] > 0)) -> use_cost[2] = use_cost[1]) /\
used_length[11][2] = 0 /\
((assignments[1][1] = 2) -> used_length[12][1] = used_length
  [11][1]) /\
((assignments[1][1] = 2) -> used_length[12][2] = (used_length
  [11][2] + cut_pieces[1][1])) /\
((not (assignments[1][1] = 2)) -> used_length[12] = used_length
  [11]) /\
((assignments[1][2] = 2) -> used_length[13][1] = used_length
  [12][1]) /\
((assignments[1][2] = 2) -> used_length[13][2] = (used_length
  [12][2] + cut_pieces[1][2])) /\
((not (assignments[1][2] = 2)) -> used_length[13] = used_length
  [12]) /\
((assignments[1][3] = 2) -> used_length[14][1] = used_length
  [13][1]) /\
((assignments[1][3] = 2) -> used_length[14][2] = (used_length
  [13][2] + cut_pieces[1][3])) /\
((not (assignments[1][3] = 2)) -> used_length[14] = used_length
  [13]) /\
((assignments[1][4] = 2) -> used_length[15][1] = used_length
  [14][1]) /\
((assignments[1][4] = 2) -> used_length[15][2] = (used_length
  [14][2] + cut_pieces[1][4])) /\
((not (assignments[1][4] = 2)) -> used_length[15] = used_length
  [14]) /\
((assignments[1][5] = 2) -> used_length[16][1] = used_length
  [15][1]) /\
((assignments[1][5] = 2) -> used_length[16][2] = (used_length
  [15][2] + cut_pieces[1][5])) /\
((not (assignments[1][5] = 2)) -> used_length[16] = used_length
  [15]) /\
((assignments[1][6] = 2) -> used_length[17][1] = used_length
  [16][1]) /\
((assignments[1][6] = 2) -> used_length[17][2] = (used_length
  [16][2] + cut_pieces[1][6])) /\
((not (assignments[1][6] = 2)) -> used_length[17] = used_length
  [16]) /\
((assignments[1][7] = 2) -> used_length[18][1] = used_length
  [17][1]) /\
((assignments[1][7] = 2) -> used_length[18][2] = (used_length
  [17][2] + cut_pieces[1][7])) /\
((not (assignments[1][7] = 2)) -> used_length[18] = used_length
  [17]) /\
((assignments[1][8] = 2) -> used_length[19][1] = used_length
  [18][1]) /\

```

```

((assignments[1][8] = 2) -> used_length[19][2] = (used_length
  [18][2] + cut_pieces[1][8])) /\
((not (assignments[1][8] = 2)) -> used_length[19] = used_length
  [18]) /\
((assignments[1][9] = 2) -> used_length[20][1] = used_length
  [19][1]) /\
((assignments[1][9] = 2) -> used_length[20][2] = (used_length
  [19][2] + cut_pieces[1][9])) /\
((not (assignments[1][9] = 2)) -> used_length[20] = used_length
  [19]) /\
((assignments[1][10] = 2) -> used_length[21][1] = used_length
  [20][1]) /\
((assignments[1][10] = 2) -> used_length[21][2] = (used_length
  [20][2] + cut_pieces[1][10])) /\
((not (assignments[1][10] = 2)) -> used_length[21] = used_length
  [20]) /\
(used_length[21][2] <= LAYER_LENGTH) /\
((used_length[21][2] > 0) -> use_cost[3] = (use_cost[2] + 3)) /\
((not (used_length[21][2] > 0)) -> use_cost[3] = use_cost[2]) /\
used_length[21][3] = 0 /\
((assignments[1][1] = 3) -> used_length[22][1] = used_length
  [21][1]) /\
((assignments[1][1] = 3) -> used_length[22][2] = used_length
  [21][2]) /\
((assignments[1][1] = 3) -> used_length[22][3] = (used_length
  [21][3] + cut_pieces[1][1])) /\
((not (assignments[1][1] = 3)) -> used_length[22] = used_length
  [21]) /\
((assignments[1][2] = 3) -> used_length[23][1] = used_length
  [22][1]) /\
((assignments[1][2] = 3) -> used_length[23][2] = used_length
  [22][2]) /\
((assignments[1][2] = 3) -> used_length[23][3] = (used_length
  [22][3] + cut_pieces[1][2])) /\
((not (assignments[1][2] = 3)) -> used_length[23] = used_length
  [22]) /\
((assignments[1][3] = 3) -> used_length[24][1] = used_length
  [23][1]) /\
((assignments[1][3] = 3) -> used_length[24][2] = used_length
  [23][2]) /\
((assignments[1][3] = 3) -> used_length[24][3] = (used_length
  [23][3] + cut_pieces[1][3])) /\
((not (assignments[1][3] = 3)) -> used_length[24] = used_length
  [23]) /\
((assignments[1][4] = 3) -> used_length[25][1] = used_length
  [24][1]) /\
((assignments[1][4] = 3) -> used_length[25][2] = used_length
  [24][2]) /\
((assignments[1][4] = 3) -> used_length[25][3] = (used_length

```

```

[24][3] + cut_pieces[1][4])) /\
((not (assignments[1][4] = 3)) -> used_length[25] = used_length
[24]) /\
((assignments[1][5] = 3) -> used_length[26][1] = used_length
[25][1]) /\
((assignments[1][5] = 3) -> used_length[26][2] = used_length
[25][2]) /\
((assignments[1][5] = 3) -> used_length[26][3] = (used_length
[25][3] + cut_pieces[1][5])) /\
((not (assignments[1][5] = 3)) -> used_length[26] = used_length
[25]) /\
((assignments[1][6] = 3) -> used_length[27][1] = used_length
[26][1]) /\
((assignments[1][6] = 3) -> used_length[27][2] = used_length
[26][2]) /\
((assignments[1][6] = 3) -> used_length[27][3] = (used_length
[26][3] + cut_pieces[1][6])) /\
((not (assignments[1][6] = 3)) -> used_length[27] = used_length
[26]) /\
((assignments[1][7] = 3) -> used_length[28][1] = used_length
[27][1]) /\
((assignments[1][7] = 3) -> used_length[28][2] = used_length
[27][2]) /\
((assignments[1][7] = 3) -> used_length[28][3] = (used_length
[27][3] + cut_pieces[1][7])) /\
((not (assignments[1][7] = 3)) -> used_length[28] = used_length
[27]) /\
((assignments[1][8] = 3) -> used_length[29][1] = used_length
[28][1]) /\
((assignments[1][8] = 3) -> used_length[29][2] = used_length
[28][2]) /\
((assignments[1][8] = 3) -> used_length[29][3] = (used_length
[28][3] + cut_pieces[1][8])) /\
((not (assignments[1][8] = 3)) -> used_length[29] = used_length
[28]) /\
((assignments[1][9] = 3) -> used_length[30][1] = used_length
[29][1]) /\
((assignments[1][9] = 3) -> used_length[30][2] = used_length
[29][2]) /\
((assignments[1][9] = 3) -> used_length[30][3] = (used_length
[29][3] + cut_pieces[1][9])) /\
((not (assignments[1][9] = 3)) -> used_length[30] = used_length
[29]) /\
((assignments[1][10] = 3) -> used_length[31][1] = used_length
[30][1]) /\
((assignments[1][10] = 3) -> used_length[31][2] = used_length
[30][2]) /\
((assignments[1][10] = 3) -> used_length[31][3] = (used_length
[30][3] + cut_pieces[1][10])) /\

```

```

((not (assignments[1][10] = 3)) -> used_length[31] = used_length
[30]) /\
(used_length[31][3] <= LAYER_LENGTH) /\
((used_length[31][3] > 0) -> use_cost[4] = (use_cost[3] + 3)) /\
((not (used_length[31][3] > 0)) -> use_cost[4] = use_cost[3]) /\
used_length[31][4] = 0 /\
((assignments[1][1] = 4) -> used_length[32][1] = used_length
[31][1]) /\
((assignments[1][1] = 4) -> used_length[32][2] = used_length
[31][2]) /\
((assignments[1][1] = 4) -> used_length[32][3] = used_length
[31][3]) /\
((assignments[1][1] = 4) -> used_length[32][4] = (used_length
[31][4] + cut_pieces[1][1])) /\
((not (assignments[1][1] = 4)) -> used_length[32] = used_length
[31]) /\
((assignments[1][2] = 4) -> used_length[33][1] = used_length
[32][1]) /\
((assignments[1][2] = 4) -> used_length[33][2] = used_length
[32][2]) /\
((assignments[1][2] = 4) -> used_length[33][3] = used_length
[32][3]) /\
((assignments[1][2] = 4) -> used_length[33][4] = (used_length
[32][4] + cut_pieces[1][2])) /\
((not (assignments[1][2] = 4)) -> used_length[33] = used_length
[32]) /\
((assignments[1][3] = 4) -> used_length[34][1] = used_length
[33][1]) /\
((assignments[1][3] = 4) -> used_length[34][2] = used_length
[33][2]) /\
((assignments[1][3] = 4) -> used_length[34][3] = used_length
[33][3]) /\
((assignments[1][3] = 4) -> used_length[34][4] = (used_length
[33][4] + cut_pieces[1][3])) /\
((not (assignments[1][3] = 4)) -> used_length[34] = used_length
[33]) /\
((assignments[1][4] = 4) -> used_length[35][1] = used_length
[34][1]) /\
((assignments[1][4] = 4) -> used_length[35][2] = used_length
[34][2]) /\
((assignments[1][4] = 4) -> used_length[35][3] = used_length
[34][3]) /\
((assignments[1][4] = 4) -> used_length[35][4] = (used_length
[34][4] + cut_pieces[1][4])) /\
((not (assignments[1][4] = 4)) -> used_length[35] = used_length
[34]) /\
((assignments[1][5] = 4) -> used_length[36][1] = used_length
[35][1]) /\
((assignments[1][5] = 4) -> used_length[36][2] = used_length

```

```

[35][2]) /\
((assignments[1][5] = 4) -> used_length[36][3] = used_length
[35][3]) /\
((assignments[1][5] = 4) -> used_length[36][4] = (used_length
[35][4] + cut_pieces[1][5])) /\
((not (assignments[1][5] = 4)) -> used_length[36] = used_length
[35]) /\
((assignments[1][6] = 4) -> used_length[37][1] = used_length
[36][1]) /\
((assignments[1][6] = 4) -> used_length[37][2] = used_length
[36][2]) /\
((assignments[1][6] = 4) -> used_length[37][3] = used_length
[36][3]) /\
((assignments[1][6] = 4) -> used_length[37][4] = (used_length
[36][4] + cut_pieces[1][6])) /\
((not (assignments[1][6] = 4)) -> used_length[37] = used_length
[36]) /\
((assignments[1][7] = 4) -> used_length[38][1] = used_length
[37][1]) /\
((assignments[1][7] = 4) -> used_length[38][2] = used_length
[37][2]) /\
((assignments[1][7] = 4) -> used_length[38][3] = used_length
[37][3]) /\
((assignments[1][7] = 4) -> used_length[38][4] = (used_length
[37][4] + cut_pieces[1][7])) /\
((not (assignments[1][7] = 4)) -> used_length[38] = used_length
[37]) /\
((assignments[1][8] = 4) -> used_length[39][1] = used_length
[38][1]) /\
((assignments[1][8] = 4) -> used_length[39][2] = used_length
[38][2]) /\
((assignments[1][8] = 4) -> used_length[39][3] = used_length
[38][3]) /\
((assignments[1][8] = 4) -> used_length[39][4] = (used_length
[38][4] + cut_pieces[1][8])) /\
((not (assignments[1][8] = 4)) -> used_length[39] = used_length
[38]) /\
((assignments[1][9] = 4) -> used_length[40][1] = used_length
[39][1]) /\
((assignments[1][9] = 4) -> used_length[40][2] = used_length
[39][2]) /\
((assignments[1][9] = 4) -> used_length[40][3] = used_length
[39][3]) /\
((assignments[1][9] = 4) -> used_length[40][4] = (used_length
[39][4] + cut_pieces[1][9])) /\
((not (assignments[1][9] = 4)) -> used_length[40] = used_length
[39]) /\
((assignments[1][10] = 4) -> used_length[41][1] = used_length
[40][1]) /\

```

```

((assignments[1][10] = 4) -> used_length[41][2] = used_length
 [40][2]) /\
((assignments[1][10] = 4) -> used_length[41][3] = used_length
 [40][3]) /\
((assignments[1][10] = 4) -> used_length[41][4] = (used_length
 [40][4] + cut_pieces[1][10])) /\
((not (assignments[1][10] = 4)) -> used_length[41] = used_length
 [40]) /\
(used_length[41][4] <= LAYER_LENGTH) /\
((used_length[41][4] > 0) -> use_cost[5] = (use_cost[4] + 3)) /\
((not (used_length[41][4] > 0)) -> use_cost[5] = use_cost[4]) /\
used_length[41][5] = 0 /\
((assignments[1][1] = 5) -> used_length[42][1] = used_length
 [41][1]) /\
((assignments[1][1] = 5) -> used_length[42][2] = used_length
 [41][2]) /\
((assignments[1][1] = 5) -> used_length[42][3] = used_length
 [41][3]) /\
((assignments[1][1] = 5) -> used_length[42][4] = used_length
 [41][4]) /\
((assignments[1][1] = 5) -> used_length[42][5] = (used_length
 [41][5] + cut_pieces[1][1])) /\
((not (assignments[1][1] = 5)) -> used_length[42] = used_length
 [41]) /\
((assignments[1][2] = 5) -> used_length[43][1] = used_length
 [42][1]) /\
((assignments[1][2] = 5) -> used_length[43][2] = used_length
 [42][2]) /\
((assignments[1][2] = 5) -> used_length[43][3] = used_length
 [42][3]) /\
((assignments[1][2] = 5) -> used_length[43][4] = used_length
 [42][4]) /\
((assignments[1][2] = 5) -> used_length[43][5] = (used_length
 [42][5] + cut_pieces[1][2])) /\
((not (assignments[1][2] = 5)) -> used_length[43] = used_length
 [42]) /\
((assignments[1][3] = 5) -> used_length[44][1] = used_length
 [43][1]) /\
((assignments[1][3] = 5) -> used_length[44][2] = used_length
 [43][2]) /\
((assignments[1][3] = 5) -> used_length[44][3] = used_length
 [43][3]) /\
((assignments[1][3] = 5) -> used_length[44][4] = used_length
 [43][4]) /\
((assignments[1][3] = 5) -> used_length[44][5] = (used_length
 [43][5] + cut_pieces[1][3])) /\
((not (assignments[1][3] = 5)) -> used_length[44] = used_length
 [43]) /\
((assignments[1][4] = 5) -> used_length[45][1] = used_length

```

```

[44][1]) /\
((assignments[1][4] = 5) -> used_length[45][2] = used_length
[44][2]) /\
((assignments[1][4] = 5) -> used_length[45][3] = used_length
[44][3]) /\
((assignments[1][4] = 5) -> used_length[45][4] = used_length
[44][4]) /\
((assignments[1][4] = 5) -> used_length[45][5] = (used_length
[44][5] + cut_pieces[1][4])) /\
((not (assignments[1][4] = 5)) -> used_length[45] = used_length
[44]) /\
((assignments[1][5] = 5) -> used_length[46][1] = used_length
[45][1]) /\
((assignments[1][5] = 5) -> used_length[46][2] = used_length
[45][2]) /\
((assignments[1][5] = 5) -> used_length[46][3] = used_length
[45][3]) /\
((assignments[1][5] = 5) -> used_length[46][4] = used_length
[45][4]) /\
((assignments[1][5] = 5) -> used_length[46][5] = (used_length
[45][5] + cut_pieces[1][5])) /\
((not (assignments[1][5] = 5)) -> used_length[46] = used_length
[45]) /\
((assignments[1][6] = 5) -> used_length[47][1] = used_length
[46][1]) /\
((assignments[1][6] = 5) -> used_length[47][2] = used_length
[46][2]) /\
((assignments[1][6] = 5) -> used_length[47][3] = used_length
[46][3]) /\
((assignments[1][6] = 5) -> used_length[47][4] = used_length
[46][4]) /\
((assignments[1][6] = 5) -> used_length[47][5] = (used_length
[46][5] + cut_pieces[1][6])) /\
((not (assignments[1][6] = 5)) -> used_length[47] = used_length
[46]) /\
((assignments[1][7] = 5) -> used_length[48][1] = used_length
[47][1]) /\
((assignments[1][7] = 5) -> used_length[48][2] = used_length
[47][2]) /\
((assignments[1][7] = 5) -> used_length[48][3] = used_length
[47][3]) /\
((assignments[1][7] = 5) -> used_length[48][4] = used_length
[47][4]) /\
((assignments[1][7] = 5) -> used_length[48][5] = (used_length
[47][5] + cut_pieces[1][7])) /\
((not (assignments[1][7] = 5)) -> used_length[48] = used_length
[47]) /\
((assignments[1][8] = 5) -> used_length[49][1] = used_length
[48][1]) /\

```

```

((assignments[1][8] = 5) -> used_length[49][2] = used_length
[48][2]) /\
((assignments[1][8] = 5) -> used_length[49][3] = used_length
[48][3]) /\
((assignments[1][8] = 5) -> used_length[49][4] = used_length
[48][4]) /\
((assignments[1][8] = 5) -> used_length[49][5] = (used_length
[48][5] + cut_pieces[1][8])) /\
((not (assignments[1][8] = 5)) -> used_length[49] = used_length
[48]) /\
((assignments[1][9] = 5) -> used_length[50][1] = used_length
[49][1]) /\
((assignments[1][9] = 5) -> used_length[50][2] = used_length
[49][2]) /\
((assignments[1][9] = 5) -> used_length[50][3] = used_length
[49][3]) /\
((assignments[1][9] = 5) -> used_length[50][4] = used_length
[49][4]) /\
((assignments[1][9] = 5) -> used_length[50][5] = (used_length
[49][5] + cut_pieces[1][9])) /\
((not (assignments[1][9] = 5)) -> used_length[50] = used_length
[49]) /\
((assignments[1][10] = 5) -> used_length[51][1] = used_length
[50][1]) /\
((assignments[1][10] = 5) -> used_length[51][2] = used_length
[50][2]) /\
((assignments[1][10] = 5) -> used_length[51][3] = used_length
[50][3]) /\
((assignments[1][10] = 5) -> used_length[51][4] = used_length
[50][4]) /\
((assignments[1][10] = 5) -> used_length[51][5] = (used_length
[50][5] + cut_pieces[1][10])) /\
((not (assignments[1][10] = 5)) -> used_length[51] = used_length
[50]) /\
(used_length[51][5] <= LAYER_LENGTH) /\
((used_length[51][5] > 0) -> use_cost[6] = (use_cost[5] + 3)) /\
((not (used_length[51][5] > 0)) -> use_cost[6] = use_cost[5]) /\
used_length[51][6] = 0 /\
((assignments[1][1] = 6) -> used_length[52][1] = used_length
[51][1]) /\
((assignments[1][1] = 6) -> used_length[52][2] = used_length
[51][2]) /\
((assignments[1][1] = 6) -> used_length[52][3] = used_length
[51][3]) /\
((assignments[1][1] = 6) -> used_length[52][4] = used_length
[51][4]) /\
((assignments[1][1] = 6) -> used_length[52][5] = used_length
[51][5]) /\
((assignments[1][1] = 6) -> used_length[52][6] = (used_length

```

```

    [51][6] + cut_pieces[1][1])) /\
((not (assignments[1][1] = 6)) -> used_length[52] = used_length
  [51]) /\
((assignments[1][2] = 6) -> used_length[53][1] = used_length
  [52][1]) /\
((assignments[1][2] = 6) -> used_length[53][2] = used_length
  [52][2]) /\
((assignments[1][2] = 6) -> used_length[53][3] = used_length
  [52][3]) /\
((assignments[1][2] = 6) -> used_length[53][4] = used_length
  [52][4]) /\
((assignments[1][2] = 6) -> used_length[53][5] = used_length
  [52][5]) /\
((assignments[1][2] = 6) -> used_length[53][6] = (used_length
  [52][6] + cut_pieces[1][2])) /\
((not (assignments[1][2] = 6)) -> used_length[53] = used_length
  [52]) /\
((assignments[1][3] = 6) -> used_length[54][1] = used_length
  [53][1]) /\
((assignments[1][3] = 6) -> used_length[54][2] = used_length
  [53][2]) /\
((assignments[1][3] = 6) -> used_length[54][3] = used_length
  [53][3]) /\
((assignments[1][3] = 6) -> used_length[54][4] = used_length
  [53][4]) /\
((assignments[1][3] = 6) -> used_length[54][5] = used_length
  [53][5]) /\
((assignments[1][3] = 6) -> used_length[54][6] = (used_length
  [53][6] + cut_pieces[1][3])) /\
((not (assignments[1][3] = 6)) -> used_length[54] = used_length
  [53]) /\
((assignments[1][4] = 6) -> used_length[55][1] = used_length
  [54][1]) /\
((assignments[1][4] = 6) -> used_length[55][2] = used_length
  [54][2]) /\
((assignments[1][4] = 6) -> used_length[55][3] = used_length
  [54][3]) /\
((assignments[1][4] = 6) -> used_length[55][4] = used_length
  [54][4]) /\
((assignments[1][4] = 6) -> used_length[55][5] = used_length
  [54][5]) /\
((assignments[1][4] = 6) -> used_length[55][6] = (used_length
  [54][6] + cut_pieces[1][4])) /\
((not (assignments[1][4] = 6)) -> used_length[55] = used_length
  [54]) /\
((assignments[1][5] = 6) -> used_length[56][1] = used_length
  [55][1]) /\
((assignments[1][5] = 6) -> used_length[56][2] = used_length
  [55][2]) /\

```

```

((assignments[1][5] = 6) -> used_length[56][3] = used_length
[55][3]) /\
((assignments[1][5] = 6) -> used_length[56][4] = used_length
[55][4]) /\
((assignments[1][5] = 6) -> used_length[56][5] = used_length
[55][5]) /\
((assignments[1][5] = 6) -> used_length[56][6] = (used_length
[55][6] + cut_pieces[1][5])) /\
((not (assignments[1][5] = 6)) -> used_length[56] = used_length
[55]) /\
((assignments[1][6] = 6) -> used_length[57][1] = used_length
[56][1]) /\
((assignments[1][6] = 6) -> used_length[57][2] = used_length
[56][2]) /\
((assignments[1][6] = 6) -> used_length[57][3] = used_length
[56][3]) /\
((assignments[1][6] = 6) -> used_length[57][4] = used_length
[56][4]) /\
((assignments[1][6] = 6) -> used_length[57][5] = used_length
[56][5]) /\
((assignments[1][6] = 6) -> used_length[57][6] = (used_length
[56][6] + cut_pieces[1][6])) /\
((not (assignments[1][6] = 6)) -> used_length[57] = used_length
[56]) /\
((assignments[1][7] = 6) -> used_length[58][1] = used_length
[57][1]) /\
((assignments[1][7] = 6) -> used_length[58][2] = used_length
[57][2]) /\
((assignments[1][7] = 6) -> used_length[58][3] = used_length
[57][3]) /\
((assignments[1][7] = 6) -> used_length[58][4] = used_length
[57][4]) /\
((assignments[1][7] = 6) -> used_length[58][5] = used_length
[57][5]) /\
((assignments[1][7] = 6) -> used_length[58][6] = (used_length
[57][6] + cut_pieces[1][7])) /\
((not (assignments[1][7] = 6)) -> used_length[58] = used_length
[57]) /\
((assignments[1][8] = 6) -> used_length[59][1] = used_length
[58][1]) /\
((assignments[1][8] = 6) -> used_length[59][2] = used_length
[58][2]) /\
((assignments[1][8] = 6) -> used_length[59][3] = used_length
[58][3]) /\
((assignments[1][8] = 6) -> used_length[59][4] = used_length
[58][4]) /\
((assignments[1][8] = 6) -> used_length[59][5] = used_length
[58][5]) /\
((assignments[1][8] = 6) -> used_length[59][6] = (used_length

```

```

    [58][6] + cut_pieces[1][8])) /\
((not (assignments[1][8] = 6)) -> used_length[59] = used_length
[58]) /\
((assignments[1][9] = 6) -> used_length[60][1] = used_length
[59][1]) /\
((assignments[1][9] = 6) -> used_length[60][2] = used_length
[59][2]) /\
((assignments[1][9] = 6) -> used_length[60][3] = used_length
[59][3]) /\
((assignments[1][9] = 6) -> used_length[60][4] = used_length
[59][4]) /\
((assignments[1][9] = 6) -> used_length[60][5] = used_length
[59][5]) /\
((assignments[1][9] = 6) -> used_length[60][6] = (used_length
[59][6] + cut_pieces[1][9])) /\
((not (assignments[1][9] = 6)) -> used_length[60] = used_length
[59]) /\
((assignments[1][10] = 6) -> used_length[61][1] = used_length
[60][1]) /\
((assignments[1][10] = 6) -> used_length[61][2] = used_length
[60][2]) /\
((assignments[1][10] = 6) -> used_length[61][3] = used_length
[60][3]) /\
((assignments[1][10] = 6) -> used_length[61][4] = used_length
[60][4]) /\
((assignments[1][10] = 6) -> used_length[61][5] = used_length
[60][5]) /\
((assignments[1][10] = 6) -> used_length[61][6] = (used_length
[60][6] + cut_pieces[1][10])) /\
((not (assignments[1][10] = 6)) -> used_length[61] = used_length
[60]) /\
(used_length[61][6] <= LAYER_LENGTH) /\
((used_length[61][6] > 0) -> use_cost[7] = (use_cost[6] + 3)) /\
((not (used_length[61][6] > 0)) -> use_cost[7] = use_cost[6]) /\
used_length[61][7] = 0 /\
((assignments[1][1] = 7) -> used_length[62][1] = used_length
[61][1]) /\
((assignments[1][1] = 7) -> used_length[62][2] = used_length
[61][2]) /\
((assignments[1][1] = 7) -> used_length[62][3] = used_length
[61][3]) /\
((assignments[1][1] = 7) -> used_length[62][4] = used_length
[61][4]) /\
((assignments[1][1] = 7) -> used_length[62][5] = used_length
[61][5]) /\
((assignments[1][1] = 7) -> used_length[62][6] = used_length
[61][6]) /\
((assignments[1][1] = 7) -> used_length[62][7] = (used_length
[61][7] + cut_pieces[1][1])) /\

```

```

((not (assignments[1][1] = 7)) -> used_length[62] = used_length
[61]) /\
((assignments[1][2] = 7) -> used_length[63][1] = used_length
[62][1]) /\
((assignments[1][2] = 7) -> used_length[63][2] = used_length
[62][2]) /\
((assignments[1][2] = 7) -> used_length[63][3] = used_length
[62][3]) /\
((assignments[1][2] = 7) -> used_length[63][4] = used_length
[62][4]) /\
((assignments[1][2] = 7) -> used_length[63][5] = used_length
[62][5]) /\
((assignments[1][2] = 7) -> used_length[63][6] = used_length
[62][6]) /\
((assignments[1][2] = 7) -> used_length[63][7] = (used_length
[62][7] + cut_pieces[1][2])) /\
((not (assignments[1][2] = 7)) -> used_length[63] = used_length
[62]) /\
((assignments[1][3] = 7) -> used_length[64][1] = used_length
[63][1]) /\
((assignments[1][3] = 7) -> used_length[64][2] = used_length
[63][2]) /\
((assignments[1][3] = 7) -> used_length[64][3] = used_length
[63][3]) /\
((assignments[1][3] = 7) -> used_length[64][4] = used_length
[63][4]) /\
((assignments[1][3] = 7) -> used_length[64][5] = used_length
[63][5]) /\
((assignments[1][3] = 7) -> used_length[64][6] = used_length
[63][6]) /\
((assignments[1][3] = 7) -> used_length[64][7] = (used_length
[63][7] + cut_pieces[1][3])) /\
((not (assignments[1][3] = 7)) -> used_length[64] = used_length
[63]) /\
((assignments[1][4] = 7) -> used_length[65][1] = used_length
[64][1]) /\
((assignments[1][4] = 7) -> used_length[65][2] = used_length
[64][2]) /\
((assignments[1][4] = 7) -> used_length[65][3] = used_length
[64][3]) /\
((assignments[1][4] = 7) -> used_length[65][4] = used_length
[64][4]) /\
((assignments[1][4] = 7) -> used_length[65][5] = used_length
[64][5]) /\
((assignments[1][4] = 7) -> used_length[65][6] = used_length
[64][6]) /\
((assignments[1][4] = 7) -> used_length[65][7] = (used_length
[64][7] + cut_pieces[1][4])) /\
((not (assignments[1][4] = 7)) -> used_length[65] = used_length

```

```

[64]) /\
((assignments[1][5] = 7) -> used_length[66][1] = used_length
[65][1]) /\
((assignments[1][5] = 7) -> used_length[66][2] = used_length
[65][2]) /\
((assignments[1][5] = 7) -> used_length[66][3] = used_length
[65][3]) /\
((assignments[1][5] = 7) -> used_length[66][4] = used_length
[65][4]) /\
((assignments[1][5] = 7) -> used_length[66][5] = used_length
[65][5]) /\
((assignments[1][5] = 7) -> used_length[66][6] = used_length
[65][6]) /\
((assignments[1][5] = 7) -> used_length[66][7] = (used_length
[65][7] + cut_pieces[1][5])) /\
((not (assignments[1][5] = 7)) -> used_length[66] = used_length
[65]) /\
((assignments[1][6] = 7) -> used_length[67][1] = used_length
[66][1]) /\
((assignments[1][6] = 7) -> used_length[67][2] = used_length
[66][2]) /\
((assignments[1][6] = 7) -> used_length[67][3] = used_length
[66][3]) /\
((assignments[1][6] = 7) -> used_length[67][4] = used_length
[66][4]) /\
((assignments[1][6] = 7) -> used_length[67][5] = used_length
[66][5]) /\
((assignments[1][6] = 7) -> used_length[67][6] = used_length
[66][6]) /\
((assignments[1][6] = 7) -> used_length[67][7] = (used_length
[66][7] + cut_pieces[1][6])) /\
((not (assignments[1][6] = 7)) -> used_length[67] = used_length
[66]) /\
((assignments[1][7] = 7) -> used_length[68][1] = used_length
[67][1]) /\
((assignments[1][7] = 7) -> used_length[68][2] = used_length
[67][2]) /\
((assignments[1][7] = 7) -> used_length[68][3] = used_length
[67][3]) /\
((assignments[1][7] = 7) -> used_length[68][4] = used_length
[67][4]) /\
((assignments[1][7] = 7) -> used_length[68][5] = used_length
[67][5]) /\
((assignments[1][7] = 7) -> used_length[68][6] = used_length
[67][6]) /\
((assignments[1][7] = 7) -> used_length[68][7] = (used_length
[67][7] + cut_pieces[1][7])) /\
((not (assignments[1][7] = 7)) -> used_length[68] = used_length
[67]) /\

```

```

((assignments[1][8] = 7) -> used_length[69][1] = used_length
[68][1]) /\
((assignments[1][8] = 7) -> used_length[69][2] = used_length
[68][2]) /\
((assignments[1][8] = 7) -> used_length[69][3] = used_length
[68][3]) /\
((assignments[1][8] = 7) -> used_length[69][4] = used_length
[68][4]) /\
((assignments[1][8] = 7) -> used_length[69][5] = used_length
[68][5]) /\
((assignments[1][8] = 7) -> used_length[69][6] = used_length
[68][6]) /\
((assignments[1][8] = 7) -> used_length[69][7] = (used_length
[68][7] + cut_pieces[1][8])) /\
((not (assignments[1][8] = 7)) -> used_length[69] = used_length
[68]) /\
((assignments[1][9] = 7) -> used_length[70][1] = used_length
[69][1]) /\
((assignments[1][9] = 7) -> used_length[70][2] = used_length
[69][2]) /\
((assignments[1][9] = 7) -> used_length[70][3] = used_length
[69][3]) /\
((assignments[1][9] = 7) -> used_length[70][4] = used_length
[69][4]) /\
((assignments[1][9] = 7) -> used_length[70][5] = used_length
[69][5]) /\
((assignments[1][9] = 7) -> used_length[70][6] = used_length
[69][6]) /\
((assignments[1][9] = 7) -> used_length[70][7] = (used_length
[69][7] + cut_pieces[1][9])) /\
((not (assignments[1][9] = 7)) -> used_length[70] = used_length
[69]) /\
((assignments[1][10] = 7) -> used_length[71][1] = used_length
[70][1]) /\
((assignments[1][10] = 7) -> used_length[71][2] = used_length
[70][2]) /\
((assignments[1][10] = 7) -> used_length[71][3] = used_length
[70][3]) /\
((assignments[1][10] = 7) -> used_length[71][4] = used_length
[70][4]) /\
((assignments[1][10] = 7) -> used_length[71][5] = used_length
[70][5]) /\
((assignments[1][10] = 7) -> used_length[71][6] = used_length
[70][6]) /\
((assignments[1][10] = 7) -> used_length[71][7] = (used_length
[70][7] + cut_pieces[1][10])) /\
((not (assignments[1][10] = 7)) -> used_length[71] = used_length
[70]) /\
(used_length[71][7] <= LAYER_LENGTH) /\

```

```

((used_length[71][7] > 0) -> use_cost[8] = (use_cost[7] + 3)) /\
((not (used_length[71][7] > 0)) -> use_cost[8] = use_cost[7]) /\
used_length[71][8] = 0 /\
((assignments[1][1] = 8) -> used_length[72][1] = used_length
[71][1]) /\
((assignments[1][1] = 8) -> used_length[72][2] = used_length
[71][2]) /\
((assignments[1][1] = 8) -> used_length[72][3] = used_length
[71][3]) /\
((assignments[1][1] = 8) -> used_length[72][4] = used_length
[71][4]) /\
((assignments[1][1] = 8) -> used_length[72][5] = used_length
[71][5]) /\
((assignments[1][1] = 8) -> used_length[72][6] = used_length
[71][6]) /\
((assignments[1][1] = 8) -> used_length[72][7] = used_length
[71][7]) /\
((assignments[1][1] = 8) -> used_length[72][8] = (used_length
[71][8] + cut_pieces[1][1])) /\
((not (assignments[1][1] = 8)) -> used_length[72] = used_length
[71]) /\
((assignments[1][2] = 8) -> used_length[73][1] = used_length
[72][1]) /\
((assignments[1][2] = 8) -> used_length[73][2] = used_length
[72][2]) /\
((assignments[1][2] = 8) -> used_length[73][3] = used_length
[72][3]) /\
((assignments[1][2] = 8) -> used_length[73][4] = used_length
[72][4]) /\
((assignments[1][2] = 8) -> used_length[73][5] = used_length
[72][5]) /\
((assignments[1][2] = 8) -> used_length[73][6] = used_length
[72][6]) /\
((assignments[1][2] = 8) -> used_length[73][7] = used_length
[72][7]) /\
((assignments[1][2] = 8) -> used_length[73][8] = (used_length
[72][8] + cut_pieces[1][2])) /\
((not (assignments[1][2] = 8)) -> used_length[73] = used_length
[72]) /\
((assignments[1][3] = 8) -> used_length[74][1] = used_length
[73][1]) /\
((assignments[1][3] = 8) -> used_length[74][2] = used_length
[73][2]) /\
((assignments[1][3] = 8) -> used_length[74][3] = used_length
[73][3]) /\
((assignments[1][3] = 8) -> used_length[74][4] = used_length
[73][4]) /\
((assignments[1][3] = 8) -> used_length[74][5] = used_length
[73][5]) /\

```

```

((assignments[1][3] = 8) -> used_length[74][6] = used_length
 [73][6]) /\
((assignments[1][3] = 8) -> used_length[74][7] = used_length
 [73][7]) /\
((assignments[1][3] = 8) -> used_length[74][8] = (used_length
 [73][8] + cut_pieces[1][3])) /\
((not (assignments[1][3] = 8)) -> used_length[74] = used_length
 [73]) /\
((assignments[1][4] = 8) -> used_length[75][1] = used_length
 [74][1]) /\
((assignments[1][4] = 8) -> used_length[75][2] = used_length
 [74][2]) /\
((assignments[1][4] = 8) -> used_length[75][3] = used_length
 [74][3]) /\
((assignments[1][4] = 8) -> used_length[75][4] = used_length
 [74][4]) /\
((assignments[1][4] = 8) -> used_length[75][5] = used_length
 [74][5]) /\
((assignments[1][4] = 8) -> used_length[75][6] = used_length
 [74][6]) /\
((assignments[1][4] = 8) -> used_length[75][7] = used_length
 [74][7]) /\
((assignments[1][4] = 8) -> used_length[75][8] = (used_length
 [74][8] + cut_pieces[1][4])) /\
((not (assignments[1][4] = 8)) -> used_length[75] = used_length
 [74]) /\
((assignments[1][5] = 8) -> used_length[76][1] = used_length
 [75][1]) /\
((assignments[1][5] = 8) -> used_length[76][2] = used_length
 [75][2]) /\
((assignments[1][5] = 8) -> used_length[76][3] = used_length
 [75][3]) /\
((assignments[1][5] = 8) -> used_length[76][4] = used_length
 [75][4]) /\
((assignments[1][5] = 8) -> used_length[76][5] = used_length
 [75][5]) /\
((assignments[1][5] = 8) -> used_length[76][6] = used_length
 [75][6]) /\
((assignments[1][5] = 8) -> used_length[76][7] = used_length
 [75][7]) /\
((assignments[1][5] = 8) -> used_length[76][8] = (used_length
 [75][8] + cut_pieces[1][5])) /\
((not (assignments[1][5] = 8)) -> used_length[76] = used_length
 [75]) /\
((assignments[1][6] = 8) -> used_length[77][1] = used_length
 [76][1]) /\
((assignments[1][6] = 8) -> used_length[77][2] = used_length
 [76][2]) /\
((assignments[1][6] = 8) -> used_length[77][3] = used_length

```

```

[76][3]) /\
((assignments[1][6] = 8) -> used_length[77][4] = used_length
[76][4]) /\
((assignments[1][6] = 8) -> used_length[77][5] = used_length
[76][5]) /\
((assignments[1][6] = 8) -> used_length[77][6] = used_length
[76][6]) /\
((assignments[1][6] = 8) -> used_length[77][7] = used_length
[76][7]) /\
((assignments[1][6] = 8) -> used_length[77][8] = (used_length
[76][8] + cut_pieces[1][6])) /\
((not (assignments[1][6] = 8)) -> used_length[77] = used_length
[76]) /\
((assignments[1][7] = 8) -> used_length[78][1] = used_length
[77][1]) /\
((assignments[1][7] = 8) -> used_length[78][2] = used_length
[77][2]) /\
((assignments[1][7] = 8) -> used_length[78][3] = used_length
[77][3]) /\
((assignments[1][7] = 8) -> used_length[78][4] = used_length
[77][4]) /\
((assignments[1][7] = 8) -> used_length[78][5] = used_length
[77][5]) /\
((assignments[1][7] = 8) -> used_length[78][6] = used_length
[77][6]) /\
((assignments[1][7] = 8) -> used_length[78][7] = used_length
[77][7]) /\
((assignments[1][7] = 8) -> used_length[78][8] = (used_length
[77][8] + cut_pieces[1][7])) /\
((not (assignments[1][7] = 8)) -> used_length[78] = used_length
[77]) /\
((assignments[1][8] = 8) -> used_length[79][1] = used_length
[78][1]) /\
((assignments[1][8] = 8) -> used_length[79][2] = used_length
[78][2]) /\
((assignments[1][8] = 8) -> used_length[79][3] = used_length
[78][3]) /\
((assignments[1][8] = 8) -> used_length[79][4] = used_length
[78][4]) /\
((assignments[1][8] = 8) -> used_length[79][5] = used_length
[78][5]) /\
((assignments[1][8] = 8) -> used_length[79][6] = used_length
[78][6]) /\
((assignments[1][8] = 8) -> used_length[79][7] = used_length
[78][7]) /\
((assignments[1][8] = 8) -> used_length[79][8] = (used_length
[78][8] + cut_pieces[1][8])) /\
((not (assignments[1][8] = 8)) -> used_length[79] = used_length
[78]) /\

```

```

((assignments[1][9] = 8) -> used_length[80][1] = used_length
[79][1]) /\
((assignments[1][9] = 8) -> used_length[80][2] = used_length
[79][2]) /\
((assignments[1][9] = 8) -> used_length[80][3] = used_length
[79][3]) /\
((assignments[1][9] = 8) -> used_length[80][4] = used_length
[79][4]) /\
((assignments[1][9] = 8) -> used_length[80][5] = used_length
[79][5]) /\
((assignments[1][9] = 8) -> used_length[80][6] = used_length
[79][6]) /\
((assignments[1][9] = 8) -> used_length[80][7] = used_length
[79][7]) /\
((assignments[1][9] = 8) -> used_length[80][8] = (used_length
[79][8] + cut_pieces[1][9])) /\
((not (assignments[1][9] = 8)) -> used_length[80] = used_length
[79]) /\
((assignments[1][10] = 8) -> used_length[81][1] = used_length
[80][1]) /\
((assignments[1][10] = 8) -> used_length[81][2] = used_length
[80][2]) /\
((assignments[1][10] = 8) -> used_length[81][3] = used_length
[80][3]) /\
((assignments[1][10] = 8) -> used_length[81][4] = used_length
[80][4]) /\
((assignments[1][10] = 8) -> used_length[81][5] = used_length
[80][5]) /\
((assignments[1][10] = 8) -> used_length[81][6] = used_length
[80][6]) /\
((assignments[1][10] = 8) -> used_length[81][7] = used_length
[80][7]) /\
((assignments[1][10] = 8) -> used_length[81][8] = (used_length
[80][8] + cut_pieces[1][10])) /\
((not (assignments[1][10] = 8)) -> used_length[81] = used_length
[80]) /\
(used_length[81][8] <= LAYER_LENGTH) /\
((used_length[81][8] > 0) -> use_cost[9] = (use_cost[8] + 3)) /\
((not (used_length[81][8] > 0)) -> use_cost[9] = use_cost[8]) /\
used_length[81][9] = 0 /\
((assignments[1][1] = 9) -> used_length[82][1] = used_length
[81][1]) /\
((assignments[1][1] = 9) -> used_length[82][2] = used_length
[81][2]) /\
((assignments[1][1] = 9) -> used_length[82][3] = used_length
[81][3]) /\
((assignments[1][1] = 9) -> used_length[82][4] = used_length
[81][4]) /\
((assignments[1][1] = 9) -> used_length[82][5] = used_length

```

```

[81][5]) /\
((assignments[1][1] = 9) -> used_length[82][6] = used_length
[81][6]) /\
((assignments[1][1] = 9) -> used_length[82][7] = used_length
[81][7]) /\
((assignments[1][1] = 9) -> used_length[82][8] = used_length
[81][8]) /\
((assignments[1][1] = 9) -> used_length[82][9] = (used_length
[81][9] + cut_pieces[1][1])) /\
((not (assignments[1][1] = 9)) -> used_length[82] = used_length
[81]) /\
((assignments[1][2] = 9) -> used_length[83][1] = used_length
[82][1]) /\
((assignments[1][2] = 9) -> used_length[83][2] = used_length
[82][2]) /\
((assignments[1][2] = 9) -> used_length[83][3] = used_length
[82][3]) /\
((assignments[1][2] = 9) -> used_length[83][4] = used_length
[82][4]) /\
((assignments[1][2] = 9) -> used_length[83][5] = used_length
[82][5]) /\
((assignments[1][2] = 9) -> used_length[83][6] = used_length
[82][6]) /\
((assignments[1][2] = 9) -> used_length[83][7] = used_length
[82][7]) /\
((assignments[1][2] = 9) -> used_length[83][8] = used_length
[82][8]) /\
((assignments[1][2] = 9) -> used_length[83][9] = (used_length
[82][9] + cut_pieces[1][2])) /\
((not (assignments[1][2] = 9)) -> used_length[83] = used_length
[82]) /\
((assignments[1][3] = 9) -> used_length[84][1] = used_length
[83][1]) /\
((assignments[1][3] = 9) -> used_length[84][2] = used_length
[83][2]) /\
((assignments[1][3] = 9) -> used_length[84][3] = used_length
[83][3]) /\
((assignments[1][3] = 9) -> used_length[84][4] = used_length
[83][4]) /\
((assignments[1][3] = 9) -> used_length[84][5] = used_length
[83][5]) /\
((assignments[1][3] = 9) -> used_length[84][6] = used_length
[83][6]) /\
((assignments[1][3] = 9) -> used_length[84][7] = used_length
[83][7]) /\
((assignments[1][3] = 9) -> used_length[84][8] = used_length
[83][8]) /\
((assignments[1][3] = 9) -> used_length[84][9] = (used_length
[83][9] + cut_pieces[1][3])) /\

```

```

((not (assignments[1][3] = 9)) -> used_length[84] = used_length
[83]) /\
((assignments[1][4] = 9) -> used_length[85][1] = used_length
[84][1]) /\
((assignments[1][4] = 9) -> used_length[85][2] = used_length
[84][2]) /\
((assignments[1][4] = 9) -> used_length[85][3] = used_length
[84][3]) /\
((assignments[1][4] = 9) -> used_length[85][4] = used_length
[84][4]) /\
((assignments[1][4] = 9) -> used_length[85][5] = used_length
[84][5]) /\
((assignments[1][4] = 9) -> used_length[85][6] = used_length
[84][6]) /\
((assignments[1][4] = 9) -> used_length[85][7] = used_length
[84][7]) /\
((assignments[1][4] = 9) -> used_length[85][8] = used_length
[84][8]) /\
((assignments[1][4] = 9) -> used_length[85][9] = (used_length
[84][9] + cut_pieces[1][4])) /\
((not (assignments[1][4] = 9)) -> used_length[85] = used_length
[84]) /\
((assignments[1][5] = 9) -> used_length[86][1] = used_length
[85][1]) /\
((assignments[1][5] = 9) -> used_length[86][2] = used_length
[85][2]) /\
((assignments[1][5] = 9) -> used_length[86][3] = used_length
[85][3]) /\
((assignments[1][5] = 9) -> used_length[86][4] = used_length
[85][4]) /\
((assignments[1][5] = 9) -> used_length[86][5] = used_length
[85][5]) /\
((assignments[1][5] = 9) -> used_length[86][6] = used_length
[85][6]) /\
((assignments[1][5] = 9) -> used_length[86][7] = used_length
[85][7]) /\
((assignments[1][5] = 9) -> used_length[86][8] = used_length
[85][8]) /\
((assignments[1][5] = 9) -> used_length[86][9] = (used_length
[85][9] + cut_pieces[1][5])) /\
((not (assignments[1][5] = 9)) -> used_length[86] = used_length
[85]) /\
((assignments[1][6] = 9) -> used_length[87][1] = used_length
[86][1]) /\
((assignments[1][6] = 9) -> used_length[87][2] = used_length
[86][2]) /\
((assignments[1][6] = 9) -> used_length[87][3] = used_length
[86][3]) /\
((assignments[1][6] = 9) -> used_length[87][4] = used_length

```

```

[86][4]) /\
((assignments[1][6] = 9) -> used_length[87][5] = used_length
[86][5]) /\
((assignments[1][6] = 9) -> used_length[87][6] = used_length
[86][6]) /\
((assignments[1][6] = 9) -> used_length[87][7] = used_length
[86][7]) /\
((assignments[1][6] = 9) -> used_length[87][8] = used_length
[86][8]) /\
((assignments[1][6] = 9) -> used_length[87][9] = (used_length
[86][9] + cut_pieces[1][6])) /\
((not (assignments[1][6] = 9)) -> used_length[87] = used_length
[86]) /\
((assignments[1][7] = 9) -> used_length[88][1] = used_length
[87][1]) /\
((assignments[1][7] = 9) -> used_length[88][2] = used_length
[87][2]) /\
((assignments[1][7] = 9) -> used_length[88][3] = used_length
[87][3]) /\
((assignments[1][7] = 9) -> used_length[88][4] = used_length
[87][4]) /\
((assignments[1][7] = 9) -> used_length[88][5] = used_length
[87][5]) /\
((assignments[1][7] = 9) -> used_length[88][6] = used_length
[87][6]) /\
((assignments[1][7] = 9) -> used_length[88][7] = used_length
[87][7]) /\
((assignments[1][7] = 9) -> used_length[88][8] = used_length
[87][8]) /\
((assignments[1][7] = 9) -> used_length[88][9] = (used_length
[87][9] + cut_pieces[1][7])) /\
((not (assignments[1][7] = 9)) -> used_length[88] = used_length
[87]) /\
((assignments[1][8] = 9) -> used_length[89][1] = used_length
[88][1]) /\
((assignments[1][8] = 9) -> used_length[89][2] = used_length
[88][2]) /\
((assignments[1][8] = 9) -> used_length[89][3] = used_length
[88][3]) /\
((assignments[1][8] = 9) -> used_length[89][4] = used_length
[88][4]) /\
((assignments[1][8] = 9) -> used_length[89][5] = used_length
[88][5]) /\
((assignments[1][8] = 9) -> used_length[89][6] = used_length
[88][6]) /\
((assignments[1][8] = 9) -> used_length[89][7] = used_length
[88][7]) /\
((assignments[1][8] = 9) -> used_length[89][8] = used_length
[88][8]) /\

```

```

((assignments[1][8] = 9) -> used_length[89][9] = (used_length
  [88][9] + cut_pieces[1][8])) /\
((not (assignments[1][8] = 9)) -> used_length[89] = used_length
  [88]) /\
((assignments[1][9] = 9) -> used_length[90][1] = used_length
  [89][1]) /\
((assignments[1][9] = 9) -> used_length[90][2] = used_length
  [89][2]) /\
((assignments[1][9] = 9) -> used_length[90][3] = used_length
  [89][3]) /\
((assignments[1][9] = 9) -> used_length[90][4] = used_length
  [89][4]) /\
((assignments[1][9] = 9) -> used_length[90][5] = used_length
  [89][5]) /\
((assignments[1][9] = 9) -> used_length[90][6] = used_length
  [89][6]) /\
((assignments[1][9] = 9) -> used_length[90][7] = used_length
  [89][7]) /\
((assignments[1][9] = 9) -> used_length[90][8] = used_length
  [89][8]) /\
((assignments[1][9] = 9) -> used_length[90][9] = (used_length
  [89][9] + cut_pieces[1][9])) /\
((not (assignments[1][9] = 9)) -> used_length[90] = used_length
  [89]) /\
((assignments[1][10] = 9) -> used_length[91][1] = used_length
  [90][1]) /\
((assignments[1][10] = 9) -> used_length[91][2] = used_length
  [90][2]) /\
((assignments[1][10] = 9) -> used_length[91][3] = used_length
  [90][3]) /\
((assignments[1][10] = 9) -> used_length[91][4] = used_length
  [90][4]) /\
((assignments[1][10] = 9) -> used_length[91][5] = used_length
  [90][5]) /\
((assignments[1][10] = 9) -> used_length[91][6] = used_length
  [90][6]) /\
((assignments[1][10] = 9) -> used_length[91][7] = used_length
  [90][7]) /\
((assignments[1][10] = 9) -> used_length[91][8] = used_length
  [90][8]) /\
((assignments[1][10] = 9) -> used_length[91][9] = (used_length
  [90][9] + cut_pieces[1][10])) /\
((not (assignments[1][10] = 9)) -> used_length[91] = used_length
  [90]) /\
(used_length[91][9] <= LAYER_LENGTH) /\
((used_length[91][9] > 0) -> use_cost[10] = (use_cost[9] + 3)) /\
((not (used_length[91][9] > 0)) -> use_cost[10] = use_cost[9]) /\
used_length[91][10] = 0 /\
((assignments[1][1] = 10) -> used_length[92][1] = used_length

```

```

[91][1]) /\
((assignments[1][1] = 10) -> used_length[92][2] = used_length
[91][2]) /\
((assignments[1][1] = 10) -> used_length[92][3] = used_length
[91][3]) /\
((assignments[1][1] = 10) -> used_length[92][4] = used_length
[91][4]) /\
((assignments[1][1] = 10) -> used_length[92][5] = used_length
[91][5]) /\
((assignments[1][1] = 10) -> used_length[92][6] = used_length
[91][6]) /\
((assignments[1][1] = 10) -> used_length[92][7] = used_length
[91][7]) /\
((assignments[1][1] = 10) -> used_length[92][8] = used_length
[91][8]) /\
((assignments[1][1] = 10) -> used_length[92][9] = used_length
[91][9]) /\
((assignments[1][1] = 10) -> used_length[92][10] = (used_length
[91][10] + cut_pieces[1][1])) /\
((not (assignments[1][1] = 10)) -> used_length[92] = used_length
[91]) /\
((assignments[1][2] = 10) -> used_length[93][1] = used_length
[92][1]) /\
((assignments[1][2] = 10) -> used_length[93][2] = used_length
[92][2]) /\
((assignments[1][2] = 10) -> used_length[93][3] = used_length
[92][3]) /\
((assignments[1][2] = 10) -> used_length[93][4] = used_length
[92][4]) /\
((assignments[1][2] = 10) -> used_length[93][5] = used_length
[92][5]) /\
((assignments[1][2] = 10) -> used_length[93][6] = used_length
[92][6]) /\
((assignments[1][2] = 10) -> used_length[93][7] = used_length
[92][7]) /\
((assignments[1][2] = 10) -> used_length[93][8] = used_length
[92][8]) /\
((assignments[1][2] = 10) -> used_length[93][9] = used_length
[92][9]) /\
((assignments[1][2] = 10) -> used_length[93][10] = (used_length
[92][10] + cut_pieces[1][2])) /\
((not (assignments[1][2] = 10)) -> used_length[93] = used_length
[92]) /\
((assignments[1][3] = 10) -> used_length[94][1] = used_length
[93][1]) /\
((assignments[1][3] = 10) -> used_length[94][2] = used_length
[93][2]) /\
((assignments[1][3] = 10) -> used_length[94][3] = used_length
[93][3]) /\

```

```

((assignments[1][3] = 10) -> used_length[94][4] = used_length
 [93][4]) /\
((assignments[1][3] = 10) -> used_length[94][5] = used_length
 [93][5]) /\
((assignments[1][3] = 10) -> used_length[94][6] = used_length
 [93][6]) /\
((assignments[1][3] = 10) -> used_length[94][7] = used_length
 [93][7]) /\
((assignments[1][3] = 10) -> used_length[94][8] = used_length
 [93][8]) /\
((assignments[1][3] = 10) -> used_length[94][9] = used_length
 [93][9]) /\
((assignments[1][3] = 10) -> used_length[94][10] = (used_length
 [93][10] + cut_pieces[1][3])) /\
((not (assignments[1][3] = 10)) -> used_length[94] = used_length
 [93]) /\
((assignments[1][4] = 10) -> used_length[95][1] = used_length
 [94][1]) /\
((assignments[1][4] = 10) -> used_length[95][2] = used_length
 [94][2]) /\
((assignments[1][4] = 10) -> used_length[95][3] = used_length
 [94][3]) /\
((assignments[1][4] = 10) -> used_length[95][4] = used_length
 [94][4]) /\
((assignments[1][4] = 10) -> used_length[95][5] = used_length
 [94][5]) /\
((assignments[1][4] = 10) -> used_length[95][6] = used_length
 [94][6]) /\
((assignments[1][4] = 10) -> used_length[95][7] = used_length
 [94][7]) /\
((assignments[1][4] = 10) -> used_length[95][8] = used_length
 [94][8]) /\
((assignments[1][4] = 10) -> used_length[95][9] = used_length
 [94][9]) /\
((assignments[1][4] = 10) -> used_length[95][10] = (used_length
 [94][10] + cut_pieces[1][4])) /\
((not (assignments[1][4] = 10)) -> used_length[95] = used_length
 [94]) /\
((assignments[1][5] = 10) -> used_length[96][1] = used_length
 [95][1]) /\
((assignments[1][5] = 10) -> used_length[96][2] = used_length
 [95][2]) /\
((assignments[1][5] = 10) -> used_length[96][3] = used_length
 [95][3]) /\
((assignments[1][5] = 10) -> used_length[96][4] = used_length
 [95][4]) /\
((assignments[1][5] = 10) -> used_length[96][5] = used_length
 [95][5]) /\
((assignments[1][5] = 10) -> used_length[96][6] = used_length

```

```

[95][6]) /\
((assignments[1][5] = 10) -> used_length[96][7] = used_length
[95][7]) /\
((assignments[1][5] = 10) -> used_length[96][8] = used_length
[95][8]) /\
((assignments[1][5] = 10) -> used_length[96][9] = used_length
[95][9]) /\
((assignments[1][5] = 10) -> used_length[96][10] = (used_length
[95][10] + cut_pieces[1][5])) /\
((not (assignments[1][5] = 10)) -> used_length[96] = used_length
[95]) /\
((assignments[1][6] = 10) -> used_length[97][1] = used_length
[96][1]) /\
((assignments[1][6] = 10) -> used_length[97][2] = used_length
[96][2]) /\
((assignments[1][6] = 10) -> used_length[97][3] = used_length
[96][3]) /\
((assignments[1][6] = 10) -> used_length[97][4] = used_length
[96][4]) /\
((assignments[1][6] = 10) -> used_length[97][5] = used_length
[96][5]) /\
((assignments[1][6] = 10) -> used_length[97][6] = used_length
[96][6]) /\
((assignments[1][6] = 10) -> used_length[97][7] = used_length
[96][7]) /\
((assignments[1][6] = 10) -> used_length[97][8] = used_length
[96][8]) /\
((assignments[1][6] = 10) -> used_length[97][9] = used_length
[96][9]) /\
((assignments[1][6] = 10) -> used_length[97][10] = (used_length
[96][10] + cut_pieces[1][6])) /\
((not (assignments[1][6] = 10)) -> used_length[97] = used_length
[96]) /\
((assignments[1][7] = 10) -> used_length[98][1] = used_length
[97][1]) /\
((assignments[1][7] = 10) -> used_length[98][2] = used_length
[97][2]) /\
((assignments[1][7] = 10) -> used_length[98][3] = used_length
[97][3]) /\
((assignments[1][7] = 10) -> used_length[98][4] = used_length
[97][4]) /\
((assignments[1][7] = 10) -> used_length[98][5] = used_length
[97][5]) /\
((assignments[1][7] = 10) -> used_length[98][6] = used_length
[97][6]) /\
((assignments[1][7] = 10) -> used_length[98][7] = used_length
[97][7]) /\
((assignments[1][7] = 10) -> used_length[98][8] = used_length
[97][8]) /\

```

```

((assignments[1][7] = 10) -> used_length[98][9] = used_length
 [97][9]) /\
((assignments[1][7] = 10) -> used_length[98][10] = (used_length
 [97][10] + cut_pieces[1][7])) /\
((not (assignments[1][7] = 10)) -> used_length[98] = used_length
 [97]) /\
((assignments[1][8] = 10) -> used_length[99][1] = used_length
 [98][1]) /\
((assignments[1][8] = 10) -> used_length[99][2] = used_length
 [98][2]) /\
((assignments[1][8] = 10) -> used_length[99][3] = used_length
 [98][3]) /\
((assignments[1][8] = 10) -> used_length[99][4] = used_length
 [98][4]) /\
((assignments[1][8] = 10) -> used_length[99][5] = used_length
 [98][5]) /\
((assignments[1][8] = 10) -> used_length[99][6] = used_length
 [98][6]) /\
((assignments[1][8] = 10) -> used_length[99][7] = used_length
 [98][7]) /\
((assignments[1][8] = 10) -> used_length[99][8] = used_length
 [98][8]) /\
((assignments[1][8] = 10) -> used_length[99][9] = used_length
 [98][9]) /\
((assignments[1][8] = 10) -> used_length[99][10] = (used_length
 [98][10] + cut_pieces[1][8])) /\
((not (assignments[1][8] = 10)) -> used_length[99] = used_length
 [98]) /\
((assignments[1][9] = 10) -> used_length[100][1] = used_length
 [99][1]) /\
((assignments[1][9] = 10) -> used_length[100][2] = used_length
 [99][2]) /\
((assignments[1][9] = 10) -> used_length[100][3] = used_length
 [99][3]) /\
((assignments[1][9] = 10) -> used_length[100][4] = used_length
 [99][4]) /\
((assignments[1][9] = 10) -> used_length[100][5] = used_length
 [99][5]) /\
((assignments[1][9] = 10) -> used_length[100][6] = used_length
 [99][6]) /\
((assignments[1][9] = 10) -> used_length[100][7] = used_length
 [99][7]) /\
((assignments[1][9] = 10) -> used_length[100][8] = used_length
 [99][8]) /\
((assignments[1][9] = 10) -> used_length[100][9] = used_length
 [99][9]) /\
((assignments[1][9] = 10) -> used_length[100][10] = (used_length
 [99][10] + cut_pieces[1][9])) /\
((not (assignments[1][9] = 10)) -> used_length[100] = used_length

```

```

    [99]) /\
((assignments[1][10] = 10) -> used_length[101][1] = used_length
  [100][1]) /\
((assignments[1][10] = 10) -> used_length[101][2] = used_length
  [100][2]) /\
((assignments[1][10] = 10) -> used_length[101][3] = used_length
  [100][3]) /\
((assignments[1][10] = 10) -> used_length[101][4] = used_length
  [100][4]) /\
((assignments[1][10] = 10) -> used_length[101][5] = used_length
  [100][5]) /\
((assignments[1][10] = 10) -> used_length[101][6] = used_length
  [100][6]) /\
((assignments[1][10] = 10) -> used_length[101][7] = used_length
  [100][7]) /\
((assignments[1][10] = 10) -> used_length[101][8] = used_length
  [100][8]) /\
((assignments[1][10] = 10) -> used_length[101][9] = used_length
  [100][9]) /\
((assignments[1][10] = 10) -> used_length[101][10] = (used_length
  [100][10] + cut_pieces[1][10])) /\
((not (assignments[1][10] = 10)) -> used_length[101] =
  used_length[100]) /\
(used_length[101][10] <= LAYER_LENGTH) /\
((used_length[101][10] > 0) -> use_cost[11] = (use_cost[10] + 3))
  /\
((not (used_length[101][10] > 0)) -> use_cost[11] = use_cost[10])
  /\
output_1 = use_cost[11]
);
int: LAYER_LENGTH = 17;
int: N_BOARDS = 5;
array[1..5] of int: ORIGINAL_BOARDS = [12, 15, 13, 10, 15];
int: MAX_BOARD_LENGTH = 15;
int: MAX_N_LAYERS = 10;
array[1..1] of array[1..5] of var 0..MAX_BOARD_LENGTH: CutPositions;
array[1..1] of array[1..10] of var 0..MAX_BOARD_LENGTH: cut_pieces;
array[1..1] of var 0..N_BOARDS: cutting_cost;
array[1..11] of array[1..10] of var 0..MAX_BOARD_LENGTH:
  cut_pieces__cutting_stage__1;
array[1..1] of array[1..5] of var 0..MAX_BOARD_LENGTH:
  cut_positions__cutting_stage__1;
array[1..6] of var 0..N_BOARDS: cutting_cost__cutting_stage__1;
array[1..1] of array[1..10] of var 1..MAX_N_LAYERS: Assignments;
array[1..2] of var 0..(N_BOARDS * 6): use_cost;
array[1..1] of array[1..10] of var 1..MAX_N_LAYERS:
  assignments__packing_stage__1;
array[1..1] of array[1..10] of var 0..MAX_BOARD_LENGTH:
  cut_pieces__packing_stage__1;

```

```
array[1..11] of var 0..(N_BOARDS * 6): use_cost__packing_stage__1;
array[1..101] of array[1..10] of var 0..sum(ORIGINAL_BOARDS):
  used_length__packing_stage__1;
array[1..1] of var 0..(N_BOARDS * 7): total_cost;
constraint cutting_stage(CutPositions[1], cut_pieces[1], cutting_cost
  [1], cut_pieces__cutting_stage__1, cut_positions__cutting_stage__1
  , cutting_cost__cutting_stage__1);
constraint use_cost[1] = 0;
constraint packing_stage(Assignments[1], cut_pieces[1], use_cost[2],
  assignments__packing_stage__1, cut_pieces__packing_stage__1,
  use_cost__packing_stage__1, used_length__packing_stage__1);
constraint total_cost[1] = (cutting_cost[1] + use_cost[2]);
solve minimize total_cost[1];
```

B Full Industrial Problem

This section presents the complete `OptDSL` specification of the industrial wood-processing problem. For clarity, the model is decomposed into constants, data structures, given instances, the overall pipeline, and the individual machine definitions. This organisation makes explicit how the full process is built from a sequence of modular transformations and checks.

We first define the global constants that determine the size of the instance and the main production constraints, such as the number of boards, maximum number of cuts, beam dimensions, and forbidden intervals.

Listing 39: Global constants for the industrial wood-processing problem in `OptDSL`

```
code_constants = f"""
N_BOARDS : int = 2
MAX_BOARD_LENGTH : int = 30
MAX_N_INTERVALS : int = 2
# Maximum number of bad (or curved) intervals per board
MAX_N_CUTS_PER_BOARD : int = 4
# Maximum number of cuts per board (including the two fixed cuts at
  the start and end of the board)

N_PIECES : int = N_BOARDS * (MAX_N_CUTS_PER_BOARD - 1)
# Total number of pieces to be obtained from all boards
BEAM_LENGTH : int = 10
# Length of the beams to be produced
BEAM_DEPTH : int = 5
# Number of layers of pieces in each beam
MAX_PIECES_PER_BEAM : int = 2
# Maximum number of pieces per beam layer
MIN_DIST_BETWEEN_PIECES : int = 1
FORBIDDEN_INTERVALS : DSLList(2, DSLList(2, int)) = [[3,4], [7,8]]
"""
```

Next, we define the overall production pipeline. This listing shows how the initial boards pass through the successive stages of reordering, cutting, filtering, reordering of pieces, and final checking, with waste minimisation as the optimisation objective.

Listing 40: Overall OptDSL pipeline for the industrial process

```
code_pipeline = f"""
initial_boards: DSLList(N_BOARDS, Board) = GIVEN_INITIAL_BOARDS

# Swapping boards

swapping_decisions_boards: DSLList(N_BOARDS - 1, bool)
swapped_boards: DSLList(N_BOARDS, Board)
swapped_boards = reordering_board_machine(initial_boards,
    swapping_decisions_boards)

# Cutting boards

cuts_list_list: DSLList(N_BOARDS, CutList)
pieces : DSLList(N_PIECES, Piece)
pieces = cutting_machine(swapped_boards, cuts_list_list)

# Filtering pieces

keep_decisions: DSLList(N_PIECES, bool)
filtered_pieces: DSLList(N_PIECES, Piece)
waste: int
filtered_pieces, waste = filtering_machine(pieces, keep_decisions)

# Reordering pieces

swapping_decisions: DSLList(N_PIECES - 1, bool)
reordered_pieces: DSLList(N_PIECES, Piece)
reordered_pieces = reordering_piece_machine(filtered_pieces,
    swapping_decisions)

# Checking pieces

pieces : DSLList(N_PIECES, Piece) = GIVEN_PIECES
checking_machine(reordered_pieces)

minimize(waste)
"""
```

To support this formulation, we introduce the record types used throughout the model. These define the structure of intervals, boards, pieces, and cut lists.

Listing 41: Type definitions used in the industrial OptDSL model

```
code_objects = ""

Interval = DSRecord({
  "start": DSInt(0, MAX_BOARD_LENGTH),
  "end": DSInt(0, MAX_BOARD_LENGTH)
})

Board = DSRecord({
  "length": DSInt(0, MAX_BOARD_LENGTH),
  "bad_intervals": DSList(MAX_N_INTERVALS, Interval),
  "curved_intervals": DSList(MAX_N_INTERVALS, Interval)
})

Piece = DSRecord({
  "length": DSInt(0, MAX_BOARD_LENGTH),
  "quality": DSInt(0, 1)
})

CutList = DSRecord({
  "position_list": DSList(MAX_N_CUTS_PER_BOARD, DSInt(0,
    MAX_BOARD_LENGTH))
})

""
```

We then specify the given input objects. These provide the initial boards and the reference list of pieces used in the example instance.

Listing 42: Example input data for boards and pieces in OPTDSL

```
code_given_objects = ""
GIVEN_INITIAL_BOARDS : DSLList(N_BOARDS, Board) = [
  Board(length=20,
    bad_intervals=[
      Interval(5,6),
      Interval(15,16)
    ],
    curved_intervals=[
      Interval(10,12),
      Interval(18,20)
    ]
  ),
  Board(length=25,
    bad_intervals=[
      Interval(10,14),
      Interval(18,20)
    ],
    curved_intervals=[
      Interval(7,9),
      Interval(18,20)
    ],
  )
]
GIVEN_PIECES : DSLList(N_PIECES, Piece) = [
  Piece(length=5, quality=1),
  Piece(length=10, quality=1),
]
""
```

The following listing defines the two reordering machines. One acts on pieces and the other on boards, both using local swap decisions to permute adjacent elements.

Listing 43: Reordering machines for boards and pieces in OPTDSL

```
code_reordering_machine = f"""
def reordering_piece_machine(list_to_reorder: DSLList(N_PIECES, Piece)
    ,
                                swapping_decisions: DSLList(N_PIECES - 1,
                                                                bool),
                                ) -> DSLList(N_PIECES, Piece):
    aux_piece : Piece
    new_list : DSLList(N_PIECES, Piece) = list_to_reorder
    for i in range(N_PIECES - 1): # Number of swapping decisions
        if swapping_decisions[i]:
            aux_piece = new_list[i]
            new_list[i] = new_list[i + 1]
            new_list[i + 1] = aux_piece
    return new_list

def reordering_board_machine(list_to_reorder: DSLList(N_BOARDS, Board)
    ,
                                swapping_decisions: DSLList(N_BOARDS - 1,
                                                                bool),
                                ) -> DSLList(N_BOARDS, Board):
    aux_board : Board
    new_list : DSLList(N_BOARDS, Board) = list_to_reorder
    for i in range(N_BOARDS - 1): # Number of swapping decisions
        if swapping_decisions[i]:
            aux_board = new_list[i]
            new_list[i] = new_list[i + 1]
            new_list[i + 1] = aux_board
    return new_list
"""
```

After reordering, boards are processed by the cutting machine. This stage generates pieces according to the proposed cuts and determines their quality of the produced pieces.

Listing 44: Cutting machine for generating pieces from boards in OPTDSL

```
code_cutting_machine = """
def cutting_machine(board_list: DSLList(N_BOARDS, Board),
                   cuts_list_list: DSLList(N_BOARDS, CutList)):
    pieces: DSLList(N_PIECES, Piece)
    quality: DSBool()
    for board_index, board in enumerate(board_list):
        cut_list : DSLList(MAX_N_CUTS_PER_BOARD, DSInt(0,
            MAX_BOARD_LENGTH)) = cuts_list_list[board_index].
            position_list
        assert cut_list[1] == 0
        assert cut_list[MAX_N_CUTS_PER_BOARD] == board.length
        curved_intervals_board : DSLList(MAX_N_INTERVALS, Interval) =
            board.curved_intervals
        for interval in curved_intervals_board:
            assert any(interval.start <= cut_list[cut] and cut_list[
                cut] <= interval.end for cut in range(1,
                    MAX_N_CUTS_PER_BOARD + 1))
        for cut_index in range(2, MAX_N_CUTS_PER_BOARD + 1):
            # Impose ordered cuts
            piece_length = cut_list[cut_index]-cut_list[cut_index-1]
            assert piece_length >= 0
            if piece_length == 0:
                assert cut_list[cut_index] == board.length
                # No piece of length 0 unless there is no more length
                to cut
                # This reduces the spaces of solutions
                # But also removes valid solutions if there is more
                than one reordering machine (unless the filtering
                machine puts all the empty pieces at the end,
                instead of just replacing discarded pieces with
                empty pieces)
            bad_intervals_board : DSLList(MAX_N_INTERVALS, Interval) =
                board.bad_intervals
            if all(
                interval.start >= cut_list[cut_index] or
                cut_list[cut_index - 1] >= interval.end
                for interval in bad_intervals_board):
                # The piece does not overlap any bad interval
                quality = 1
            else:
                quality = 0
            pieces[(board_index - 1) * (MAX_N_CUTS_PER_BOARD - 1) + (
                cut_index - 1)] = {"quality": quality, "length":
                piece_length}
    return pieces"""
```

The filtering machine removes unusable pieces and accumulates the corresponding waste. Discarded pieces are replaced by empty pieces so that the list length remains fixed.

Listing 45: Filtering machine for selecting valid pieces and computing waste in OPTDSL

```
code_filtering_machine = """
def filtering_machine(list_to_filter: DSLList(N_PIECES, Piece),
                    keep_decisions: DSLList(N_PIECES, bool),
                    ):
    waste : int = 0
    # Initialize filtered list with empty pieces
    filtered_list : DSLList(N_PIECES, Piece)
    for i in range(N_PIECES):
        if keep_decisions[i]:
            assert list_to_filter[i].quality == 1
            filtered_list[i] = list_to_filter[i]
        else:
            waste = waste + list_to_filter[i].length
            filtered_list[i] = {"quality": 1, "length": 0}

    return filtered_list, waste
"""
```

Finally, the checking machine verifies whether the reordered pieces can be assembled into beams satisfying the geometric and spacing constraints of the industrial process.

Listing 46: Checking machine for validating beam assembly constraints in OPTDSL

```
code_check_machine = """
def checking_machine(pieces: DSLList(N_PIECES, elem_type = Piece)):
    depth = 0
    length = 0
    n_length = 0
    n_prev_layer = 0
    new_beam = 1
    current_index = 0
    for piece in pieces:
        current_index = current_index + 1
        length = length + piece.length
        n_length = n_length + 1
        assert length <= BEAM_LENGTH
        if length == BEAM_LENGTH:
            depth = depth + 1
            n_prev_layer = n_length
            n_length = 0
            length = 0
            if depth == BEAM_DEPTH:
                new_beam = 1
                depth = 0
        else:
            new_beam = 0
            for i in range(1, MAX_PIECES_PER_BEAM):
                if i < n_prev_layer:
                    start = current_index - n_length - n_prev_layer +
                        1
                    end = start + i - 1
                    s = 0
                    for j in range(1, MAX_PIECES_PER_BEAM):
                        # the language can be made more elegant
                        # taking in for from i to j, and
                        # automatically translating into this if
                        # lower and upper bounds are known
                        if start <= j:
                            if j <= end:
                                s = s + pieces[j].length
                                assert abs(s - length) >=
                                    MIN_DIST_BETWEEN_PIECES
                    # Check forbidden intervals
                    for interval in FORBIDDEN_INTERVALS:
                        assert not (interval[1] <= length and length <=
                            interval[2])
    """
```

The generated MiniZinc translation of this OptDSL code using our code is:

Listing 47: Generated MiniZinc

```
type Interval = record(  
  0..MAX_BOARD_LENGTH: start,  
  0..MAX_BOARD_LENGTH: end  
);  
type Board = record(  
  0..MAX_BOARD_LENGTH: length,  
  array[1..2] of Interval: bad_intervals,  
  array[1..2] of Interval: curved_intervals  
);  
type Piece = record(  
  0..MAX_BOARD_LENGTH: length,  
  0..1: quality  
);  
type CutList = record(  
  array[1..4] of 0..MAX_BOARD_LENGTH: position_list  
);  
predicate checking_machine(array[1..6] of record(  
  var 0..MAX_BOARD_LENGTH: length,  
  var 0..1: quality  
): input_1, array[1..7] of var int: current_index, array[1..13] of  
var int: depth, array[1..1] of var int: end, array[1..13] of var  
int: length, array[1..13] of var int: n_length, array[1..7] of var  
int: n_prev_layer, array[1..7] of var int: new_beam, array[1..1]  
of array[1..6] of record(  
  var 0..MAX_BOARD_LENGTH: length,  
  var 0..1: quality  
): pieces, array[1..7] of var int: s, array[1..1] of var int: start)  
=  
(  
  pieces[1][1].length = input_1[1].length /\  
  pieces[1][1].quality = input_1[1].quality /\  
  pieces[1][2].length = input_1[2].length /\  
  pieces[1][2].quality = input_1[2].quality /\  
  pieces[1][3].length = input_1[3].length /\  
  pieces[1][3].quality = input_1[3].quality /\  
  pieces[1][4].length = input_1[4].length /\  
  pieces[1][4].quality = input_1[4].quality /\  
  pieces[1][5].length = input_1[5].length /\  
  pieces[1][5].quality = input_1[5].quality /\  
  pieces[1][6].length = input_1[6].length /\  
  pieces[1][6].quality = input_1[6].quality /\  
  depth[1] = 0 /\  
  length[1] = 0 /\  
  n_length[1] = 0 /\  
  n_prev_layer[1] = 0 /\  
  new_beam[1] = 1 /\
```

```

current_index[1] = 0 /\
current_index[2] = (current_index[1] + 1) /\
length[2] = (length[1] + pieces[1][1].length) /\
n_length[2] = (n_length[1] + 1) /\
(length[2] <= BEAM_LENGTH) /\
((length[2] = BEAM_LENGTH) -> depth[2] = (depth[1] + 1)) /\
((length[2] = BEAM_LENGTH) -> n_prev_layer[2] = n_length[2]) /\
((length[2] = BEAM_LENGTH) -> n_length[3] = 0) /\
((length[2] = BEAM_LENGTH) -> length[3] = 0) /\
((depth[2] = BEAM_DEPTH) /\ (length[2] = BEAM_LENGTH) -> new_beam
  [2] = 1) /\
((depth[2] = BEAM_DEPTH) /\ (length[2] = BEAM_LENGTH) -> depth[3]
  = 0) /\
((not (depth[2] = BEAM_DEPTH)) /\ (length[2] = BEAM_LENGTH) ->
  new_beam[2] = new_beam[1]) /\
((not (depth[2] = BEAM_DEPTH)) /\ (length[2] = BEAM_LENGTH) ->
  depth[3] = depth[2]) /\
((not (length[2] = BEAM_LENGTH)) -> new_beam[2] = 0) /\
((1 < n_prev_layer[1]) /\ (not (length[2] = BEAM_LENGTH)) ->
  start[1] = (((current_index[2] - n_length[2]) - n_prev_layer
  [1]) + 1)) /\
((1 < n_prev_layer[1]) /\ (not (length[2] = BEAM_LENGTH)) -> end
  [1] = ((start[1] + 1) - 1)) /\
((1 < n_prev_layer[1]) /\ (not (length[2] = BEAM_LENGTH)) -> s[1]
  = 0) /\
((1 <= end[1]) /\ (start[1] <= 1) /\ (1 < n_prev_layer[1]) /\ (
  not (length[2] = BEAM_LENGTH)) -> s[2] = (s[1] + pieces[1][1].
  length)) /\
((1 <= end[1]) /\ (start[1] <= 1) /\ (1 < n_prev_layer[1]) /\ (
  not (length[2] = BEAM_LENGTH)) -> (abs((s[2] - length[2])) >=
  MIN_DIST_BETWEEN_PIECES)) /\
((not (1 <= end[1])) /\ (start[1] <= 1) /\ (1 < n_prev_layer[1])
  /\ (not (length[2] = BEAM_LENGTH)) -> s[2] = s[1]) /\
((not (start[1] <= 1)) /\ (1 < n_prev_layer[1]) /\ (not (length
  [2] = BEAM_LENGTH)) -> s[2] = s[1]) /\
((not (length[2] = BEAM_LENGTH)) -> (not ((FORBIDDEN_INTERVALS
  [1][1] <= length[2]) /\ (length[2] <= FORBIDDEN_INTERVALS
  [1][2])))) /\
((not (length[2] = BEAM_LENGTH)) -> (not ((FORBIDDEN_INTERVALS
  [2][1] <= length[2]) /\ (length[2] <= FORBIDDEN_INTERVALS
  [2][2])))) /\
((not (length[2] = BEAM_LENGTH)) -> length[3] = length[2]) /\
((not (length[2] = BEAM_LENGTH)) -> n_length[3] = n_length[2]) /\
((not (length[2] = BEAM_LENGTH)) -> depth[3] = depth[1]) /\
((not (length[2] = BEAM_LENGTH)) -> n_prev_layer[2] =
  n_prev_layer[1]) /\
current_index[3] = (current_index[2] + 1) /\
length[4] = (length[3] + pieces[1][2].length) /\
n_length[4] = (n_length[3] + 1) /\

```

```

(length[4] <= BEAM_LENGTH) /\
((length[4] = BEAM_LENGTH) -> depth[4] = (depth[3] + 1)) /\
((length[4] = BEAM_LENGTH) -> n_prev_layer[3] = n_length[4]) /\
((length[4] = BEAM_LENGTH) -> n_length[5] = 0) /\
((length[4] = BEAM_LENGTH) -> length[5] = 0) /\
((depth[4] = BEAM_DEPTH) /\ (length[4] = BEAM_LENGTH) -> new_beam
  [3] = 1) /\
((depth[4] = BEAM_DEPTH) /\ (length[4] = BEAM_LENGTH) -> depth[5]
  = 0) /\
((not (depth[4] = BEAM_DEPTH)) /\ (length[4] = BEAM_LENGTH) ->
  new_beam[3] = new_beam[2]) /\
((not (depth[4] = BEAM_DEPTH)) /\ (length[4] = BEAM_LENGTH) ->
  depth[5] = depth[4]) /\
((length[4] = BEAM_LENGTH) -> s[3] = s[2]) /\
((not (length[4] = BEAM_LENGTH)) -> new_beam[3] = 0) /\
((1 < n_prev_layer[2]) /\ (not (length[4] = BEAM_LENGTH)) ->
  start[1] = (((current_index[3] - n_length[4]) - n_prev_layer
  [2]) + 1)) /\
((1 < n_prev_layer[2]) /\ (not (length[4] = BEAM_LENGTH)) -> end
  [1] = ((start[1] + 1) - 1)) /\
((1 < n_prev_layer[2]) /\ (not (length[4] = BEAM_LENGTH)) -> s[2]
  = 0) /\
((1 <= end[1]) /\ (start[1] <= 1) /\ (1 < n_prev_layer[2]) /\ (
  not (length[4] = BEAM_LENGTH)) -> s[3] = (s[2] + pieces[1][1].
  length)) /\
((1 <= end[1]) /\ (start[1] <= 1) /\ (1 < n_prev_layer[2]) /\ (
  not (length[4] = BEAM_LENGTH)) -> (abs((s[3] - length[4])) >=
  MIN_DIST_BETWEEN_PIECES)) /\
((not (1 <= end[1])) /\ (start[1] <= 1) /\ (1 < n_prev_layer[2])
  /\ (not (length[4] = BEAM_LENGTH)) -> s[3] = s[2]) /\
((not (start[1] <= 1)) /\ (1 < n_prev_layer[2]) /\ (not (length
  [4] = BEAM_LENGTH)) -> s[3] = s[2]) /\
((not (1 < n_prev_layer[2])) /\ (not (length[4] = BEAM_LENGTH))
  -> s[3] = s[2]) /\
((not (length[4] = BEAM_LENGTH)) -> (not ((FORBIDDEN_INTERVALS
  [1][1] <= length[4]) /\ (length[4] <= FORBIDDEN_INTERVALS
  [1][2])))) /\
((not (length[4] = BEAM_LENGTH)) -> (not ((FORBIDDEN_INTERVALS
  [2][1] <= length[4]) /\ (length[4] <= FORBIDDEN_INTERVALS
  [2][2])))) /\
((not (length[4] = BEAM_LENGTH)) -> length[5] = length[4]) /\
((not (length[4] = BEAM_LENGTH)) -> n_length[5] = n_length[4]) /\
((not (length[4] = BEAM_LENGTH)) -> depth[5] = depth[3]) /\
((not (length[4] = BEAM_LENGTH)) -> n_prev_layer[3] =
  n_prev_layer[2]) /\
current_index[4] = (current_index[3] + 1) /\
length[6] = (length[5] + pieces[1][3].length) /\
n_length[6] = (n_length[5] + 1) /\
(length[6] <= BEAM_LENGTH) /\

```

```

((length[6] = BEAM_LENGTH) -> depth[6] = (depth[5] + 1)) /\
((length[6] = BEAM_LENGTH) -> n_prev_layer[4] = n_length[6]) /\
((length[6] = BEAM_LENGTH) -> n_length[7] = 0) /\
((length[6] = BEAM_LENGTH) -> length[7] = 0) /\
((depth[6] = BEAM_DEPTH) /\ (length[6] = BEAM_LENGTH) -> new_beam
[4] = 1) /\
((depth[6] = BEAM_DEPTH) /\ (length[6] = BEAM_LENGTH) -> depth[7]
= 0) /\
((not (depth[6] = BEAM_DEPTH)) /\ (length[6] = BEAM_LENGTH) ->
new_beam[4] = new_beam[3]) /\
((not (depth[6] = BEAM_DEPTH)) /\ (length[6] = BEAM_LENGTH) ->
depth[7] = depth[6]) /\
((length[6] = BEAM_LENGTH) -> s[4] = s[3]) /\
((not (length[6] = BEAM_LENGTH)) -> new_beam[4] = 0) /\
((1 < n_prev_layer[3]) /\ (not (length[6] = BEAM_LENGTH)) ->
start[1] = (((current_index[4] - n_length[6]) - n_prev_layer
[3]) + 1)) /\
((1 < n_prev_layer[3]) /\ (not (length[6] = BEAM_LENGTH)) -> end
[1] = ((start[1] + 1) - 1)) /\
((1 < n_prev_layer[3]) /\ (not (length[6] = BEAM_LENGTH)) -> s[3]
= 0) /\
((1 <= end[1]) /\ (start[1] <= 1) /\ (1 < n_prev_layer[3]) /\ (
not (length[6] = BEAM_LENGTH) -> s[4] = (s[3] + pieces[1][1].
length)) /\
((1 <= end[1]) /\ (start[1] <= 1) /\ (1 < n_prev_layer[3]) /\ (
not (length[6] = BEAM_LENGTH) -> (abs((s[4] - length[6])) >=
MIN_DIST_BETWEEN_PIECES)) /\
((not (1 <= end[1])) /\ (start[1] <= 1) /\ (1 < n_prev_layer[3])
/\ (not (length[6] = BEAM_LENGTH) -> s[4] = s[3]) /\
((not (start[1] <= 1)) /\ (1 < n_prev_layer[3]) /\ (not (length
[6] = BEAM_LENGTH) -> s[4] = s[3]) /\
((not (1 < n_prev_layer[3])) /\ (not (length[6] = BEAM_LENGTH))
-> s[4] = s[3]) /\
((not (length[6] = BEAM_LENGTH)) -> (not ((FORBIDDEN_INTERVALS
[1][1] <= length[6]) /\ (length[6] <= FORBIDDEN_INTERVALS
[1][2])))) /\
((not (length[6] = BEAM_LENGTH)) -> (not ((FORBIDDEN_INTERVALS
[2][1] <= length[6]) /\ (length[6] <= FORBIDDEN_INTERVALS
[2][2])))) /\
((not (length[6] = BEAM_LENGTH)) -> length[7] = length[6]) /\
((not (length[6] = BEAM_LENGTH)) -> n_length[7] = n_length[6]) /\
((not (length[6] = BEAM_LENGTH)) -> depth[7] = depth[5]) /\
((not (length[6] = BEAM_LENGTH)) -> n_prev_layer[4] =
n_prev_layer[3]) /\
current_index[5] = (current_index[4] + 1) /\
length[8] = (length[7] + pieces[1][4].length) /\
n_length[8] = (n_length[7] + 1) /\
(length[8] <= BEAM_LENGTH) /\
((length[8] = BEAM_LENGTH) -> depth[8] = (depth[7] + 1)) /\

```

```

((length[8] = BEAM_LENGTH) -> n_prev_layer[5] = n_length[8]) /\
((length[8] = BEAM_LENGTH) -> n_length[9] = 0) /\
((length[8] = BEAM_LENGTH) -> length[9] = 0) /\
((depth[8] = BEAM_DEPTH) /\ (length[8] = BEAM_LENGTH) -> new_beam
[5] = 1) /\
((depth[8] = BEAM_DEPTH) /\ (length[8] = BEAM_LENGTH) -> depth[9]
= 0) /\
((not (depth[8] = BEAM_DEPTH)) /\ (length[8] = BEAM_LENGTH) ->
new_beam[5] = new_beam[4]) /\
((not (depth[8] = BEAM_DEPTH)) /\ (length[8] = BEAM_LENGTH) ->
depth[9] = depth[8]) /\
((length[8] = BEAM_LENGTH) -> s[5] = s[4]) /\
((not (length[8] = BEAM_LENGTH)) -> new_beam[5] = 0) /\
((1 < n_prev_layer[4]) /\ (not (length[8] = BEAM_LENGTH)) ->
start[1] = (((current_index[5] - n_length[8]) - n_prev_layer
[4]) + 1)) /\
((1 < n_prev_layer[4]) /\ (not (length[8] = BEAM_LENGTH)) -> end
[1] = ((start[1] + 1) - 1)) /\
((1 < n_prev_layer[4]) /\ (not (length[8] = BEAM_LENGTH)) -> s[4]
= 0) /\
((1 <= end[1]) /\ (start[1] <= 1) /\ (1 < n_prev_layer[4]) /\ (
not (length[8] = BEAM_LENGTH)) -> s[5] = (s[4] + pieces[1][1].
length)) /\
((1 <= end[1]) /\ (start[1] <= 1) /\ (1 < n_prev_layer[4]) /\ (
not (length[8] = BEAM_LENGTH)) -> (abs((s[5] - length[8])) >=
MIN_DIST_BETWEEN_PIECES)) /\
((not (1 <= end[1])) /\ (start[1] <= 1) /\ (1 < n_prev_layer[4])
/\ (not (length[8] = BEAM_LENGTH)) -> s[5] = s[4]) /\
((not (start[1] <= 1)) /\ (1 < n_prev_layer[4]) /\ (not (length
[8] = BEAM_LENGTH)) -> s[5] = s[4]) /\
((not (1 < n_prev_layer[4])) /\ (not (length[8] = BEAM_LENGTH))
-> s[5] = s[4]) /\
((not (length[8] = BEAM_LENGTH)) -> (not ((FORBIDDEN_INTERVALS
[1][1] <= length[8]) /\ (length[8] <= FORBIDDEN_INTERVALS
[1][2])))) /\
((not (length[8] = BEAM_LENGTH)) -> (not ((FORBIDDEN_INTERVALS
[2][1] <= length[8]) /\ (length[8] <= FORBIDDEN_INTERVALS
[2][2])))) /\
((not (length[8] = BEAM_LENGTH)) -> length[9] = length[8]) /\
((not (length[8] = BEAM_LENGTH)) -> n_length[9] = n_length[8]) /\
((not (length[8] = BEAM_LENGTH)) -> depth[9] = depth[7]) /\
((not (length[8] = BEAM_LENGTH)) -> n_prev_layer[5] =
n_prev_layer[4]) /\
current_index[6] = (current_index[5] + 1) /\
length[10] = (length[9] + pieces[1][5].length) /\
n_length[10] = (n_length[9] + 1) /\
(length[10] <= BEAM_LENGTH) /\
((length[10] = BEAM_LENGTH) -> depth[10] = (depth[9] + 1)) /\
((length[10] = BEAM_LENGTH) -> n_prev_layer[6] = n_length[10]) /\

```

```

((length[10] = BEAM_LENGTH) -> n_length[11] = 0) /\
((length[10] = BEAM_LENGTH) -> length[11] = 0) /\
((depth[10] = BEAM_DEPTH) /\ (length[10] = BEAM_LENGTH) ->
  new_beam[6] = 1) /\
((depth[10] = BEAM_DEPTH) /\ (length[10] = BEAM_LENGTH) -> depth
  [11] = 0) /\
((not (depth[10] = BEAM_DEPTH)) /\ (length[10] = BEAM_LENGTH) ->
  new_beam[6] = new_beam[5]) /\
((not (depth[10] = BEAM_DEPTH)) /\ (length[10] = BEAM_LENGTH) ->
  depth[11] = depth[10]) /\
((length[10] = BEAM_LENGTH) -> s[6] = s[5]) /\
((not (length[10] = BEAM_LENGTH)) -> new_beam[6] = 0) /\
((1 < n_prev_layer[5]) /\ (not (length[10] = BEAM_LENGTH)) ->
  start[1] = (((current_index[6] - n_length[10]) - n_prev_layer
  [5]) + 1)) /\
((1 < n_prev_layer[5]) /\ (not (length[10] = BEAM_LENGTH)) -> end
  [1] = ((start[1] + 1) - 1)) /\
((1 < n_prev_layer[5]) /\ (not (length[10] = BEAM_LENGTH)) -> s
  [5] = 0) /\
((1 <= end[1]) /\ (start[1] <= 1) /\ (1 < n_prev_layer[5]) /\ (
  not (length[10] = BEAM_LENGTH)) -> s[6] = (s[5] + pieces
  [1][1].length)) /\
((1 <= end[1]) /\ (start[1] <= 1) /\ (1 < n_prev_layer[5]) /\ (
  not (length[10] = BEAM_LENGTH)) -> (abs((s[6] - length[10]))
  >= MIN_DIST_BETWEEN_PIECES)) /\
((not (1 <= end[1])) /\ (start[1] <= 1) /\ (1 < n_prev_layer[5])
  /\ (not (length[10] = BEAM_LENGTH)) -> s[6] = s[5]) /\
((not (start[1] <= 1)) /\ (1 < n_prev_layer[5]) /\ (not (length
  [10] = BEAM_LENGTH)) -> s[6] = s[5]) /\
((not (1 < n_prev_layer[5])) /\ (not (length[10] = BEAM_LENGTH))
  -> s[6] = s[5]) /\
((not (length[10] = BEAM_LENGTH)) -> (not ((FORBIDDEN_INTERVALS
  [1][1] <= length[10]) /\ (length[10] <= FORBIDDEN_INTERVALS
  [1][2])))) /\
((not (length[10] = BEAM_LENGTH)) -> (not ((FORBIDDEN_INTERVALS
  [2][1] <= length[10]) /\ (length[10] <= FORBIDDEN_INTERVALS
  [2][2])))) /\
((not (length[10] = BEAM_LENGTH)) -> length[11] = length[10]) /\
((not (length[10] = BEAM_LENGTH)) -> n_length[11] = n_length[10])
  /\
((not (length[10] = BEAM_LENGTH)) -> depth[11] = depth[9]) /\
((not (length[10] = BEAM_LENGTH)) -> n_prev_layer[6] =
  n_prev_layer[5]) /\
current_index[7] = (current_index[6] + 1) /\
length[12] = (length[11] + pieces[1][6].length) /\
n_length[12] = (n_length[11] + 1) /\
(length[12] <= BEAM_LENGTH) /\
((length[12] = BEAM_LENGTH) -> depth[12] = (depth[11] + 1)) /\
((length[12] = BEAM_LENGTH) -> n_prev_layer[7] = n_length[12]) /\

```

```

((length[12] = BEAM_LENGTH) -> n_length[13] = 0) /\
((length[12] = BEAM_LENGTH) -> length[13] = 0) /\
((depth[12] = BEAM_DEPTH) /\ (length[12] = BEAM_LENGTH) ->
  new_beam[7] = 1) /\
((depth[12] = BEAM_DEPTH) /\ (length[12] = BEAM_LENGTH) -> depth
  [13] = 0) /\
((not (depth[12] = BEAM_DEPTH)) /\ (length[12] = BEAM_LENGTH) ->
  new_beam[7] = new_beam[6]) /\
((not (depth[12] = BEAM_DEPTH)) /\ (length[12] = BEAM_LENGTH) ->
  depth[13] = depth[12]) /\
((length[12] = BEAM_LENGTH) -> s[7] = s[6]) /\
((not (length[12] = BEAM_LENGTH)) -> new_beam[7] = 0) /\
((1 < n_prev_layer[6]) /\ (not (length[12] = BEAM_LENGTH)) ->
  start[1] = (((current_index[7] - n_length[12]) - n_prev_layer
  [6]) + 1)) /\
((1 < n_prev_layer[6]) /\ (not (length[12] = BEAM_LENGTH)) -> end
  [1] = ((start[1] + 1) - 1)) /\
((1 < n_prev_layer[6]) /\ (not (length[12] = BEAM_LENGTH)) -> s
  [6] = 0) /\
((1 <= end[1]) /\ (start[1] <= 1) /\ (1 < n_prev_layer[6]) /\ (
  not (length[12] = BEAM_LENGTH)) -> s[7] = (s[6] + pieces
  [1][1].length)) /\
((1 <= end[1]) /\ (start[1] <= 1) /\ (1 < n_prev_layer[6]) /\ (
  not (length[12] = BEAM_LENGTH)) -> (abs((s[7] - length[12]))
  >= MIN_DIST_BETWEEN_PIECES)) /\
((not (1 <= end[1])) /\ (start[1] <= 1) /\ (1 < n_prev_layer[6])
  /\ (not (length[12] = BEAM_LENGTH)) -> s[7] = s[6]) /\
((not (start[1] <= 1)) /\ (1 < n_prev_layer[6]) /\ (not (length
  [12] = BEAM_LENGTH)) -> s[7] = s[6]) /\
((not (1 < n_prev_layer[6])) /\ (not (length[12] = BEAM_LENGTH))
  -> s[7] = s[6]) /\
((not (length[12] = BEAM_LENGTH)) -> (not ((FORBIDDEN_INTERVALS
  [1][1] <= length[12]) /\ (length[12] <= FORBIDDEN_INTERVALS
  [1][2])))) /\
((not (length[12] = BEAM_LENGTH)) -> (not ((FORBIDDEN_INTERVALS
  [2][1] <= length[12]) /\ (length[12] <= FORBIDDEN_INTERVALS
  [2][2])))) /\
((not (length[12] = BEAM_LENGTH)) -> length[13] = length[12]) /\
((not (length[12] = BEAM_LENGTH)) -> n_length[13] = n_length[12])
  /\
((not (length[12] = BEAM_LENGTH)) -> depth[13] = depth[11]) /\
((not (length[12] = BEAM_LENGTH)) -> n_prev_layer[7] =
  n_prev_layer[6])
);
predicate cutting_machine(array[1..2] of record(
  var 0..MAX_BOARD_LENGTH: length,
  array[1..2] of record(
    var 0..MAX_BOARD_LENGTH: start,
    var 0..MAX_BOARD_LENGTH: end

```

```

): bad_intervals,
    array[1..2] of record(
        var 0..MAX_BOARD_LENGTH: start,
        var 0..MAX_BOARD_LENGTH: end
    )
): curved_intervals
): input_1, array[1..2] of record(
    array[1..4] of var 0..MAX_BOARD_LENGTH: position_list
): input_2, array[1..6] of record(
    var 0..MAX_BOARD_LENGTH: length,
    var 0..1: quality
): output_1, array[1..6] of array[1..2] of record(
    var 0..MAX_BOARD_LENGTH: start,
    var 0..MAX_BOARD_LENGTH: end
): bad_intervals_board, array[1..1] of array[1..2] of record(
    var 0..MAX_BOARD_LENGTH: length,
    array[1..2] of record(
        var 0..MAX_BOARD_LENGTH: start,
        var 0..MAX_BOARD_LENGTH: end
    )
): bad_intervals,
    array[1..2] of record(
        var 0..MAX_BOARD_LENGTH: start,
        var 0..MAX_BOARD_LENGTH: end
    )
): curved_intervals
): board_list, array[1..2] of array[1..2] of record(
    var 0..MAX_BOARD_LENGTH: start,
    var 0..MAX_BOARD_LENGTH: end
): curved_intervals_board, array[1..2] of array[1..4] of var 0..
MAX_BOARD_LENGTH: cut_list, array[1..1] of array[1..2] of record(
    array[1..4] of var 0..MAX_BOARD_LENGTH: position_list
): cuts_list_list, array[1..6] of var int: piece_length, array[1..7]
of array[1..6] of record(
    var 0..MAX_BOARD_LENGTH: length,
    var 0..1: quality
): pieces, array[1..6] of var bool: quality) =
(
    board_list[1][1].length = input_1[1].length /\
    board_list[1][1].bad_intervals[1].start = input_1[1].
        bad_intervals[1].start /\
    board_list[1][1].bad_intervals[1].end = input_1[1].bad_intervals
        [1].end /\
    board_list[1][1].bad_intervals[2].start = input_1[1].
        bad_intervals[2].start /\
    board_list[1][1].bad_intervals[2].end = input_1[1].bad_intervals
        [2].end /\
    board_list[1][1].curved_intervals[1].start = input_1[1].
        curved_intervals[1].start /\
    board_list[1][1].curved_intervals[1].end = input_1[1].
        curved_intervals[1].end /\
    board_list[1][1].curved_intervals[2].start = input_1[1].

```

```

    curved_intervals[2].start /\
board_list[1][1].curved_intervals[2].end = input_1[1].
    curved_intervals[2].end /\
board_list[1][2].length = input_1[2].length /\
board_list[1][2].bad_intervals[1].start = input_1[2].
    bad_intervals[1].start /\
board_list[1][2].bad_intervals[1].end = input_1[2].bad_intervals
    [1].end /\
board_list[1][2].bad_intervals[2].start = input_1[2].
    bad_intervals[2].start /\
board_list[1][2].bad_intervals[2].end = input_1[2].bad_intervals
    [2].end /\
board_list[1][2].curved_intervals[1].start = input_1[2].
    curved_intervals[1].start /\
board_list[1][2].curved_intervals[1].end = input_1[2].
    curved_intervals[1].end /\
board_list[1][2].curved_intervals[2].start = input_1[2].
    curved_intervals[2].start /\
board_list[1][2].curved_intervals[2].end = input_1[2].
    curved_intervals[2].end /\
cuts_list_list[1][1].position_list[1] = input_2[1].position_list
    [1] /\
cuts_list_list[1][1].position_list[2] = input_2[1].position_list
    [2] /\
cuts_list_list[1][1].position_list[3] = input_2[1].position_list
    [3] /\
cuts_list_list[1][1].position_list[4] = input_2[1].position_list
    [4] /\
cuts_list_list[1][2].position_list[1] = input_2[2].position_list
    [1] /\
cuts_list_list[1][2].position_list[2] = input_2[2].position_list
    [2] /\
cuts_list_list[1][2].position_list[3] = input_2[2].position_list
    [3] /\
cuts_list_list[1][2].position_list[4] = input_2[2].position_list
    [4] /\
cut_list[1][1] = cuts_list_list[1][1].position_list[1] /\
cut_list[1][2] = cuts_list_list[1][1].position_list[2] /\
cut_list[1][3] = cuts_list_list[1][1].position_list[3] /\
cut_list[1][4] = cuts_list_list[1][1].position_list[4] /\
(cut_list[1][1] = 0) /\
(cut_list[1][MAX_N_CUTS_PER_BOARD] = board_list[1][1].length) /\
curved_intervals_board[1][1].start = board_list[1][1].
    curved_intervals[1].start /\
curved_intervals_board[1][1].end = board_list[1][1].
    curved_intervals[1].end /\
curved_intervals_board[1][2].start = board_list[1][1].
    curved_intervals[2].start /\
curved_intervals_board[1][2].end = board_list[1][1].

```

```

    curved_intervals[2].end /\
(((curved_intervals_board[1][1].start <= cut_list[1][1]) /\ (
  cut_list[1][1] <= curved_intervals_board[1][1].end)) \/ ((
  curved_intervals_board[1][1].start <= cut_list[1][2]) /\ (
  cut_list[1][2] <= curved_intervals_board[1][1].end)) \/ ((
  curved_intervals_board[1][1].start <= cut_list[1][3]) /\ (
  cut_list[1][3] <= curved_intervals_board[1][1].end)) \/ ((
  curved_intervals_board[1][1].start <= cut_list[1][4]) /\ (
  cut_list[1][4] <= curved_intervals_board[1][1].end))) /\
(((curved_intervals_board[1][2].start <= cut_list[1][1]) /\ (
  cut_list[1][1] <= curved_intervals_board[1][2].end)) \/ ((
  curved_intervals_board[1][2].start <= cut_list[1][2]) /\ (
  cut_list[1][2] <= curved_intervals_board[1][2].end)) \/ ((
  curved_intervals_board[1][2].start <= cut_list[1][3]) /\ (
  cut_list[1][3] <= curved_intervals_board[1][2].end)) \/ ((
  curved_intervals_board[1][2].start <= cut_list[1][4]) /\ (
  cut_list[1][4] <= curved_intervals_board[1][2].end))) /\
piece_length[1] = (cut_list[1][2] - cut_list[1][(2 - 1)]) /\
(piece_length[1] >= 0) /\
((piece_length[1] = 0) -> (cut_list[1][2] = board_list[1][1].
  length)) /\
bad_intervals_board[1][1].start = board_list[1][1].bad_intervals
  [1].start /\
bad_intervals_board[1][1].end = board_list[1][1].bad_intervals
  [1].end /\
bad_intervals_board[1][2].start = board_list[1][1].bad_intervals
  [2].start /\
bad_intervals_board[1][2].end = board_list[1][1].bad_intervals
  [2].end /\
((((bad_intervals_board[1][1].start >= cut_list[1][2]) \/ (
  cut_list[1][(2 - 1)] >= bad_intervals_board[1][1].end)) /\ ((
  bad_intervals_board[1][2].start >= cut_list[1][2]) \/ (
  cut_list[1][(2 - 1)] >= bad_intervals_board[1][2].end))) ->
  quality[1] = 1) /\
((not (((bad_intervals_board[1][1].start >= cut_list[1][2]) \/ (
  cut_list[1][(2 - 1)] >= bad_intervals_board[1][1].end)) /\ ((
  bad_intervals_board[1][2].start >= cut_list[1][2]) \/ (
  cut_list[1][(2 - 1)] >= bad_intervals_board[1][2].end)))) ->
  quality[1] = 0) /\
pieces[2][(((1 - 1) * (MAX_N_CUTS_PER_BOARD - 1)) + (2 - 1))].
  length = (quality: quality[1], length: piece_length[1]).length
  /\
pieces[2][(((1 - 1) * (MAX_N_CUTS_PER_BOARD - 1)) + (2 - 1))].
  quality = (quality: quality[1], length: piece_length[1]).
  quality /\
piece_length[2] = (cut_list[1][3] - cut_list[1][(3 - 1)]) /\
(piece_length[2] >= 0) /\
((piece_length[2] = 0) -> (cut_list[1][3] = board_list[1][1].
  length)) /\

```

```

bad_intervals_board[2][1].start = board_list[1][1].bad_intervals
    [1].start /\
bad_intervals_board[2][1].end = board_list[1][1].bad_intervals
    [1].end /\
bad_intervals_board[2][2].start = board_list[1][1].bad_intervals
    [2].start /\
bad_intervals_board[2][2].end = board_list[1][1].bad_intervals
    [2].end /\
((((bad_intervals_board[2][1].start >= cut_list[1][3]) \\/ (
    cut_list[1][(3 - 1)] >= bad_intervals_board[2][1].end)) /\ ((
    bad_intervals_board[2][2].start >= cut_list[1][3]) \\/ (
    cut_list[1][(3 - 1)] >= bad_intervals_board[2][2].end))) ->
    quality[2] = 1) /\
((not (((bad_intervals_board[2][1].start >= cut_list[1][3]) \\/ (
    cut_list[1][(3 - 1)] >= bad_intervals_board[2][1].end)) /\ ((
    bad_intervals_board[2][2].start >= cut_list[1][3]) \\/ (
    cut_list[1][(3 - 1)] >= bad_intervals_board[2][2].end)))) ->
    quality[2] = 0) /\
pieces[3][1].quality = pieces[2][1].quality /\
pieces[3][1].length = pieces[2][1].length /\
pieces[3][2].quality = pieces[2][2].quality /\
pieces[3][2].length = pieces[2][2].length /\
pieces[3][3].quality = pieces[2][3].quality /\
pieces[3][3].length = pieces[2][3].length /\
pieces[3][4].quality = pieces[2][4].quality /\
pieces[3][4].length = pieces[2][4].length /\
pieces[3][5].quality = pieces[2][5].quality /\
pieces[3][5].length = pieces[2][5].length /\
pieces[3][6].quality = pieces[2][6].quality /\
pieces[3][6].length = pieces[2][6].length /\
pieces[3][(((1 - 1) * (MAX_N_CUTS_PER_BOARD - 1)) + (3 - 1))].
    length = (quality: quality[2], length: piece_length[2]).length
    /\
pieces[3][(((1 - 1) * (MAX_N_CUTS_PER_BOARD - 1)) + (3 - 1))].
    quality = (quality: quality[2], length: piece_length[2]).
    quality /\
piece_length[3] = (cut_list[1][4] - cut_list[1][(4 - 1)]) /\
(piece_length[3] >= 0) /\
((piece_length[3] = 0) -> (cut_list[1][4] = board_list[1][1].
    length)) /\
bad_intervals_board[3][1].start = board_list[1][1].bad_intervals
    [1].start /\
bad_intervals_board[3][1].end = board_list[1][1].bad_intervals
    [1].end /\
bad_intervals_board[3][2].start = board_list[1][1].bad_intervals
    [2].start /\
bad_intervals_board[3][2].end = board_list[1][1].bad_intervals
    [2].end /\
((((bad_intervals_board[3][1].start >= cut_list[1][4]) \\/ (

```

```

cut_list[1][(4 - 1)] >= bad_intervals_board[3][1].end)) /\ ((
bad_intervals_board[3][2].start >= cut_list[1][4]) \\/ (
cut_list[1][(4 - 1)] >= bad_intervals_board[3][2].end))) ->
quality[3] = 1) /\
((not (((bad_intervals_board[3][1].start >= cut_list[1][4]) \\/ (
cut_list[1][(4 - 1)] >= bad_intervals_board[3][1].end)) /\ ((
bad_intervals_board[3][2].start >= cut_list[1][4]) \\/ (
cut_list[1][(4 - 1)] >= bad_intervals_board[3][2].end)))) ->
quality[3] = 0) /\
pieces[4][1].quality = pieces[3][1].quality /\
pieces[4][1].length = pieces[3][1].length /\
pieces[4][2].quality = pieces[3][2].quality /\
pieces[4][2].length = pieces[3][2].length /\
pieces[4][3].quality = pieces[3][3].quality /\
pieces[4][3].length = pieces[3][3].length /\
pieces[4][4].quality = pieces[3][4].quality /\
pieces[4][4].length = pieces[3][4].length /\
pieces[4][5].quality = pieces[3][5].quality /\
pieces[4][5].length = pieces[3][5].length /\
pieces[4][6].quality = pieces[3][6].quality /\
pieces[4][6].length = pieces[3][6].length /\
pieces[4][(((1 - 1) * (MAX_N_CUTS_PER_BOARD - 1)) + (4 - 1))].
length = (quality: quality[3], length: piece_length[3]).length
/\
pieces[4][(((1 - 1) * (MAX_N_CUTS_PER_BOARD - 1)) + (4 - 1))].
quality = (quality: quality[3], length: piece_length[3]).
quality /\
cut_list[2][1] = cuts_list_list[1][2].position_list[1] /\
cut_list[2][2] = cuts_list_list[1][2].position_list[2] /\
cut_list[2][3] = cuts_list_list[1][2].position_list[3] /\
cut_list[2][4] = cuts_list_list[1][2].position_list[4] /\
(cut_list[2][1] = 0) /\
(cut_list[2][MAX_N_CUTS_PER_BOARD] = board_list[1][2].length) /\
curved_intervals_board[2][1].start = board_list[1][2].
curved_intervals[1].start /\
curved_intervals_board[2][1].end = board_list[1][2].
curved_intervals[1].end /\
curved_intervals_board[2][2].start = board_list[1][2].
curved_intervals[2].start /\
curved_intervals_board[2][2].end = board_list[1][2].
curved_intervals[2].end /\
(((curved_intervals_board[2][1].start <= cut_list[2][1]) /\ (
cut_list[2][1] <= curved_intervals_board[2][1].end)) \\/ ((
curved_intervals_board[2][1].start <= cut_list[2][2]) /\ (
cut_list[2][2] <= curved_intervals_board[2][1].end)) \\/ ((
curved_intervals_board[2][1].start <= cut_list[2][3]) /\ (
cut_list[2][3] <= curved_intervals_board[2][1].end)) \\/ ((
curved_intervals_board[2][1].start <= cut_list[2][4]) /\ (
cut_list[2][4] <= curved_intervals_board[2][1].end))) /\

```

```

(((curved_intervals_board[2][2].start <= cut_list[2][1]) /\ (
  cut_list[2][1] <= curved_intervals_board[2][2].end)) \/ ((
  curved_intervals_board[2][2].start <= cut_list[2][2]) /\ (
  cut_list[2][2] <= curved_intervals_board[2][2].end)) \/ ((
  curved_intervals_board[2][2].start <= cut_list[2][3]) /\ (
  cut_list[2][3] <= curved_intervals_board[2][2].end)) \/ ((
  curved_intervals_board[2][2].start <= cut_list[2][4]) /\ (
  cut_list[2][4] <= curved_intervals_board[2][2].end))) /\
piece_length[4] = (cut_list[2][2] - cut_list[2][(2 - 1)]) /\
(piece_length[4] >= 0) /\
((piece_length[4] = 0) -> (cut_list[2][2] = board_list[1][2].
  length)) /\
bad_intervals_board[4][1].start = board_list[1][2].bad_intervals
  [1].start /\
bad_intervals_board[4][1].end = board_list[1][2].bad_intervals
  [1].end /\
bad_intervals_board[4][2].start = board_list[1][2].bad_intervals
  [2].start /\
bad_intervals_board[4][2].end = board_list[1][2].bad_intervals
  [2].end /\
((((bad_intervals_board[4][1].start >= cut_list[2][2]) \/ (
  cut_list[2][(2 - 1)] >= bad_intervals_board[4][1].end)) /\ ((
  bad_intervals_board[4][2].start >= cut_list[2][2]) \/ (
  cut_list[2][(2 - 1)] >= bad_intervals_board[4][2].end))) ->
  quality[4] = 1) /\
((not (((bad_intervals_board[4][1].start >= cut_list[2][2]) \/ (
  cut_list[2][(2 - 1)] >= bad_intervals_board[4][1].end)) /\ ((
  bad_intervals_board[4][2].start >= cut_list[2][2]) \/ (
  cut_list[2][(2 - 1)] >= bad_intervals_board[4][2].end)))) ->
  quality[4] = 0) /\
pieces[5][1].quality = pieces[4][1].quality /\
pieces[5][1].length = pieces[4][1].length /\
pieces[5][2].quality = pieces[4][2].quality /\
pieces[5][2].length = pieces[4][2].length /\
pieces[5][3].quality = pieces[4][3].quality /\
pieces[5][3].length = pieces[4][3].length /\
pieces[5][4].quality = pieces[4][4].quality /\
pieces[5][4].length = pieces[4][4].length /\
pieces[5][5].quality = pieces[4][5].quality /\
pieces[5][5].length = pieces[4][5].length /\
pieces[5][6].quality = pieces[4][6].quality /\
pieces[5][6].length = pieces[4][6].length /\
pieces[5][(((2 - 1) * (MAX_N_CUTS_PER_BOARD - 1)) + (2 - 1))].
  length = (quality: quality[4], length: piece_length[4]).length
  /\
pieces[5][(((2 - 1) * (MAX_N_CUTS_PER_BOARD - 1)) + (2 - 1))].
  quality = (quality: quality[4], length: piece_length[4]).
  quality /\
piece_length[5] = (cut_list[2][3] - cut_list[2][(3 - 1)]) /\

```

```

(piece_length[5] >= 0) /\
((piece_length[5] = 0) -> (cut_list[2][3] = board_list[1][2].
length)) /\
bad_intervals_board[5][1].start = board_list[1][2].bad_intervals
[1].start /\
bad_intervals_board[5][1].end = board_list[1][2].bad_intervals
[1].end /\
bad_intervals_board[5][2].start = board_list[1][2].bad_intervals
[2].start /\
bad_intervals_board[5][2].end = board_list[1][2].bad_intervals
[2].end /\
((((bad_intervals_board[5][1].start >= cut_list[2][3]) \\/ (
cut_list[2][(3 - 1)] >= bad_intervals_board[5][1].end)) /\ ((
bad_intervals_board[5][2].start >= cut_list[2][3]) \\/ (
cut_list[2][(3 - 1)] >= bad_intervals_board[5][2].end))) ->
quality[5] = 1) /\
((not (((bad_intervals_board[5][1].start >= cut_list[2][3]) \\/ (
cut_list[2][(3 - 1)] >= bad_intervals_board[5][1].end)) /\ ((
bad_intervals_board[5][2].start >= cut_list[2][3]) \\/ (
cut_list[2][(3 - 1)] >= bad_intervals_board[5][2].end)))) ->
quality[5] = 0) /\
pieces[6][1].quality = pieces[5][1].quality /\
pieces[6][1].length = pieces[5][1].length /\
pieces[6][2].quality = pieces[5][2].quality /\
pieces[6][2].length = pieces[5][2].length /\
pieces[6][3].quality = pieces[5][3].quality /\
pieces[6][3].length = pieces[5][3].length /\
pieces[6][4].quality = pieces[5][4].quality /\
pieces[6][4].length = pieces[5][4].length /\
pieces[6][5].quality = pieces[5][5].quality /\
pieces[6][5].length = pieces[5][5].length /\
pieces[6][6].quality = pieces[5][6].quality /\
pieces[6][6].length = pieces[5][6].length /\
pieces[6][(((2 - 1) * (MAX_N_CUTS_PER_BOARD - 1)) + (3 - 1))].
length = (quality: quality[5], length: piece_length[5]).length
/\
pieces[6][(((2 - 1) * (MAX_N_CUTS_PER_BOARD - 1)) + (3 - 1))].
quality = (quality: quality[5], length: piece_length[5]).
quality /\
piece_length[6] = (cut_list[2][4] - cut_list[2][(4 - 1)]) /\
(piece_length[6] >= 0) /\
((piece_length[6] = 0) -> (cut_list[2][4] = board_list[1][2].
length)) /\
bad_intervals_board[6][1].start = board_list[1][2].bad_intervals
[1].start /\
bad_intervals_board[6][1].end = board_list[1][2].bad_intervals
[1].end /\
bad_intervals_board[6][2].start = board_list[1][2].bad_intervals
[2].start /\

```

```

bad_intervals_board[6][2].end = board_list[1][2].bad_intervals
  [2].end /\
(((bad_intervals_board[6][1].start >= cut_list[2][4]) \\/ (
  cut_list[2][(4 - 1)] >= bad_intervals_board[6][1].end)) /\ ((
  bad_intervals_board[6][2].start >= cut_list[2][4]) \\/ (
  cut_list[2][(4 - 1)] >= bad_intervals_board[6][2].end))) ->
  quality[6] = 1) /\
((not (((bad_intervals_board[6][1].start >= cut_list[2][4]) \\/ (
  cut_list[2][(4 - 1)] >= bad_intervals_board[6][1].end)) /\ ((
  bad_intervals_board[6][2].start >= cut_list[2][4]) \\/ (
  cut_list[2][(4 - 1)] >= bad_intervals_board[6][2].end)))) ->
  quality[6] = 0) /\
pieces[7][1].quality = pieces[6][1].quality /\
pieces[7][1].length = pieces[6][1].length /\
pieces[7][2].quality = pieces[6][2].quality /\
pieces[7][2].length = pieces[6][2].length /\
pieces[7][3].quality = pieces[6][3].quality /\
pieces[7][3].length = pieces[6][3].length /\
pieces[7][4].quality = pieces[6][4].quality /\
pieces[7][4].length = pieces[6][4].length /\
pieces[7][5].quality = pieces[6][5].quality /\
pieces[7][5].length = pieces[6][5].length /\
pieces[7][6].quality = pieces[6][6].quality /\
pieces[7][6].length = pieces[6][6].length /\
pieces[7][(((2 - 1) * (MAX_N_CUTS_PER_BOARD - 1)) + (4 - 1))].
  length = (quality: quality[6], length: piece_length[6]).length
  /\
pieces[7][(((2 - 1) * (MAX_N_CUTS_PER_BOARD - 1)) + (4 - 1))].
  quality = (quality: quality[6], length: piece_length[6]).
  quality /\
output_1 = pieces[7]
);
predicate filtering_machine(array[1..6] of record(
  var 0..MAX_BOARD_LENGTH: length,
  var 0..1: quality
): input_1, array[1..6] of var bool: input_2, array[1..6] of record(
  var 0..MAX_BOARD_LENGTH: length,
  var 0..1: quality
): output_1, var int: output_2, array[1..1] of array[1..6] of record(
  var 0..MAX_BOARD_LENGTH: length,
  var 0..1: quality
): filtered_list, array[1..1] of array[1..6] of var bool:
  keep_decisions, array[1..1] of array[1..6] of record(
  var 0..MAX_BOARD_LENGTH: length,
  var 0..1: quality
): list_to_filter, array[1..6] of var int: waste) =
  (
  list_to_filter[1][1].length = input_1[1].length /\
  list_to_filter[1][1].quality = input_1[1].quality /\

```

```

list_to_filter[1][2].length = input_1[2].length /\
list_to_filter[1][2].quality = input_1[2].quality /\
list_to_filter[1][3].length = input_1[3].length /\
list_to_filter[1][3].quality = input_1[3].quality /\
list_to_filter[1][4].length = input_1[4].length /\
list_to_filter[1][4].quality = input_1[4].quality /\
list_to_filter[1][5].length = input_1[5].length /\
list_to_filter[1][5].quality = input_1[5].quality /\
list_to_filter[1][6].length = input_1[6].length /\
list_to_filter[1][6].quality = input_1[6].quality /\
keep_decisions[1][1] = input_2[1] /\
keep_decisions[1][2] = input_2[2] /\
keep_decisions[1][3] = input_2[3] /\
keep_decisions[1][4] = input_2[4] /\
keep_decisions[1][5] = input_2[5] /\
keep_decisions[1][6] = input_2[6] /\
waste[1] = 0 /\
(keep_decisions[1][1] -> (list_to_filter[1][1].quality = 1)) /\
(keep_decisions[1][1] -> filtered_list[1][1].length =
  list_to_filter[1][1].length) /\
(keep_decisions[1][1] -> filtered_list[1][1].quality =
  list_to_filter[1][1].quality) /\
(keep_decisions[1][1] -> waste[2] = waste[1]) /\
((not keep_decisions[1][1]) -> waste[2] = (waste[1] +
  list_to_filter[1][1].length)) /\
((not keep_decisions[1][1]) -> filtered_list[1][1].length = (
  quality: 1, length: 0).length) /\
((not keep_decisions[1][1]) -> filtered_list[1][1].quality = (
  quality: 1, length: 0).quality) /\
(keep_decisions[1][2] -> (list_to_filter[1][2].quality = 1)) /\
(keep_decisions[1][2] -> filtered_list[1][2].length =
  list_to_filter[1][2].length) /\
(keep_decisions[1][2] -> filtered_list[1][2].quality =
  list_to_filter[1][2].quality) /\
(keep_decisions[1][2] -> waste[3] = waste[2]) /\
((not keep_decisions[1][2]) -> waste[3] = (waste[2] +
  list_to_filter[1][2].length)) /\
((not keep_decisions[1][2]) -> filtered_list[1][2].length = (
  quality: 1, length: 0).length) /\
((not keep_decisions[1][2]) -> filtered_list[1][2].quality = (
  quality: 1, length: 0).quality) /\
(keep_decisions[1][3] -> (list_to_filter[1][3].quality = 1)) /\
(keep_decisions[1][3] -> filtered_list[1][3].length =
  list_to_filter[1][3].length) /\
(keep_decisions[1][3] -> filtered_list[1][3].quality =
  list_to_filter[1][3].quality) /\
(keep_decisions[1][3] -> waste[4] = waste[3]) /\
((not keep_decisions[1][3]) -> waste[4] = (waste[3] +
  list_to_filter[1][3].length)) /\

```

```

((not keep_decisions[1][3]) -> filtered_list[1][3].length = (
  quality: 1, length: 0).length) /\
((not keep_decisions[1][3]) -> filtered_list[1][3].quality = (
  quality: 1, length: 0).quality) /\
(keep_decisions[1][4] -> (list_to_filter[1][4].quality = 1)) /\
(keep_decisions[1][4] -> filtered_list[1][4].length =
  list_to_filter[1][4].length) /\
(keep_decisions[1][4] -> filtered_list[1][4].quality =
  list_to_filter[1][4].quality) /\
(keep_decisions[1][4] -> waste[5] = waste[4]) /\
((not keep_decisions[1][4]) -> waste[5] = (waste[4] +
  list_to_filter[1][4].length)) /\
((not keep_decisions[1][4]) -> filtered_list[1][4].length = (
  quality: 1, length: 0).length) /\
((not keep_decisions[1][4]) -> filtered_list[1][4].quality = (
  quality: 1, length: 0).quality) /\
(keep_decisions[1][5] -> (list_to_filter[1][5].quality = 1)) /\
(keep_decisions[1][5] -> filtered_list[1][5].length =
  list_to_filter[1][5].length) /\
(keep_decisions[1][5] -> filtered_list[1][5].quality =
  list_to_filter[1][5].quality) /\
(keep_decisions[1][5] -> waste[6] = waste[5]) /\
((not keep_decisions[1][5]) -> waste[6] = (waste[5] +
  list_to_filter[1][5].length)) /\
((not keep_decisions[1][5]) -> filtered_list[1][5].length = (
  quality: 1, length: 0).length) /\
((not keep_decisions[1][5]) -> filtered_list[1][5].quality = (
  quality: 1, length: 0).quality) /\
output_1 = filtered_list[1] /\
output_2 = waste[6]
);
predicate reordering_board_machine(array[1..2] of record(
  var 0..MAX_BOARD_LENGTH: length,
  array[1..2] of record(
    var 0..MAX_BOARD_LENGTH: start,
    var 0..MAX_BOARD_LENGTH: end
  ): bad_intervals,
  array[1..2] of record(
    var 0..MAX_BOARD_LENGTH: start,
    var 0..MAX_BOARD_LENGTH: end
  ): curved_intervals
): input_1, array[1..1] of var bool: input_2, array[1..2] of record(
  var 0..MAX_BOARD_LENGTH: length,
  array[1..2] of record(
    var 0..MAX_BOARD_LENGTH: start,
    var 0..MAX_BOARD_LENGTH: end
  ): bad_intervals,
  array[1..2] of record(
    var 0..MAX_BOARD_LENGTH: start,

```

```

    var 0..MAX_BOARD_LENGTH: end
): curved_intervals
): output_1, array[1..1] of record(
    var 0..MAX_BOARD_LENGTH: length,
    array[1..2] of record(
        var 0..MAX_BOARD_LENGTH: start,
        var 0..MAX_BOARD_LENGTH: end
    )
): bad_intervals,
    array[1..2] of record(
        var 0..MAX_BOARD_LENGTH: start,
        var 0..MAX_BOARD_LENGTH: end
    )
): curved_intervals
): aux_board, array[1..1] of array[1..2] of record(
    var 0..MAX_BOARD_LENGTH: length,
    array[1..2] of record(
        var 0..MAX_BOARD_LENGTH: start,
        var 0..MAX_BOARD_LENGTH: end
    )
): bad_intervals,
    array[1..2] of record(
        var 0..MAX_BOARD_LENGTH: start,
        var 0..MAX_BOARD_LENGTH: end
    )
): curved_intervals
): list_to_reorder, array[1..1] of array[1..2] of record(
    var 0..MAX_BOARD_LENGTH: length,
    array[1..2] of record(
        var 0..MAX_BOARD_LENGTH: start,
        var 0..MAX_BOARD_LENGTH: end
    )
): bad_intervals,
    array[1..2] of record(
        var 0..MAX_BOARD_LENGTH: start,
        var 0..MAX_BOARD_LENGTH: end
    )
): curved_intervals
): new_list, array[1..1] of array[1..1] of var bool:
    swapping_decisions) =
    (
    list_to_reorder[1][1].length = input_1[1].length /\
    list_to_reorder[1][1].bad_intervals[1].start = input_1[1].
        bad_intervals[1].start /\
    list_to_reorder[1][1].bad_intervals[1].end = input_1[1].
        bad_intervals[1].end /\
    list_to_reorder[1][1].bad_intervals[2].start = input_1[1].
        bad_intervals[2].start /\
    list_to_reorder[1][1].bad_intervals[2].end = input_1[1].
        bad_intervals[2].end /\
    list_to_reorder[1][1].curved_intervals[1].start = input_1[1].
        curved_intervals[1].start /\
    list_to_reorder[1][1].curved_intervals[1].end = input_1[1].
        curved_intervals[1].end /\
    list_to_reorder[1][1].curved_intervals[2].start = input_1[1].

```

```

    curved_intervals[2].start /\
list_to_reorder[1][1].curved_intervals[2].end = input_1[1].
    curved_intervals[2].end /\
list_to_reorder[1][2].length = input_1[2].length /\
list_to_reorder[1][2].bad_intervals[1].start = input_1[2].
    bad_intervals[1].start /\
list_to_reorder[1][2].bad_intervals[1].end = input_1[2].
    bad_intervals[1].end /\
list_to_reorder[1][2].bad_intervals[2].start = input_1[2].
    bad_intervals[2].start /\
list_to_reorder[1][2].bad_intervals[2].end = input_1[2].
    bad_intervals[2].end /\
list_to_reorder[1][2].curved_intervals[1].start = input_1[2].
    curved_intervals[1].start /\
list_to_reorder[1][2].curved_intervals[1].end = input_1[2].
    curved_intervals[1].end /\
list_to_reorder[1][2].curved_intervals[2].start = input_1[2].
    curved_intervals[2].start /\
list_to_reorder[1][2].curved_intervals[2].end = input_1[2].
    curved_intervals[2].end /\
swapping_decisions[1][1] = input_2[1] /\
new_list[1][1].length = list_to_reorder[1][1].length /\
new_list[1][1].bad_intervals[1].start = list_to_reorder[1][1].
    bad_intervals[1].start /\
new_list[1][1].bad_intervals[1].end = list_to_reorder[1][1].
    bad_intervals[1].end /\
new_list[1][1].bad_intervals[2].start = list_to_reorder[1][1].
    bad_intervals[2].start /\
new_list[1][1].bad_intervals[2].end = list_to_reorder[1][1].
    bad_intervals[2].end /\
new_list[1][1].curved_intervals[1].start = list_to_reorder[1][1].
    curved_intervals[1].start /\
new_list[1][1].curved_intervals[1].end = list_to_reorder[1][1].
    curved_intervals[1].end /\
new_list[1][1].curved_intervals[2].start = list_to_reorder[1][1].
    curved_intervals[2].start /\
new_list[1][1].curved_intervals[2].end = list_to_reorder[1][1].
    curved_intervals[2].end /\
new_list[1][2].length = list_to_reorder[1][2].length /\
new_list[1][2].bad_intervals[1].start = list_to_reorder[1][2].
    bad_intervals[1].start /\
new_list[1][2].bad_intervals[1].end = list_to_reorder[1][2].
    bad_intervals[1].end /\
new_list[1][2].bad_intervals[2].start = list_to_reorder[1][2].
    bad_intervals[2].start /\
new_list[1][2].bad_intervals[2].end = list_to_reorder[1][2].
    bad_intervals[2].end /\
new_list[1][2].curved_intervals[1].start = list_to_reorder[1][2].
    curved_intervals[1].start /\

```

```

new_list[1][2].curved_intervals[1].end = list_to_reorder[1][2].
  curved_intervals[1].end /\
new_list[1][2].curved_intervals[2].start = list_to_reorder[1][2].
  curved_intervals[2].start /\
new_list[1][2].curved_intervals[2].end = list_to_reorder[1][2].
  curved_intervals[2].end /\
output_1 = new_list[1]
);
predicate reordering_piece_machine(array[1..6] of record(
  var 0..MAX_BOARD_LENGTH: length,
  var 0..1: quality
): input_1, array[1..5] of var bool: input_2, array[1..6] of record(
  var 0..MAX_BOARD_LENGTH: length,
  var 0..1: quality
): output_1, array[1..4] of record(
  var 0..MAX_BOARD_LENGTH: length,
  var 0..1: quality
): aux_piece, array[1..1] of array[1..6] of record(
  var 0..MAX_BOARD_LENGTH: length,
  var 0..1: quality
): list_to_reorder, array[1..9] of array[1..6] of record(
  var 0..MAX_BOARD_LENGTH: length,
  var 0..1: quality
): new_list, array[1..1] of array[1..5] of var bool:
  swapping_decisions) =
(
list_to_reorder[1][1].length = input_1[1].length /\
list_to_reorder[1][1].quality = input_1[1].quality /\
list_to_reorder[1][2].length = input_1[2].length /\
list_to_reorder[1][2].quality = input_1[2].quality /\
list_to_reorder[1][3].length = input_1[3].length /\
list_to_reorder[1][3].quality = input_1[3].quality /\
list_to_reorder[1][4].length = input_1[4].length /\
list_to_reorder[1][4].quality = input_1[4].quality /\
list_to_reorder[1][5].length = input_1[5].length /\
list_to_reorder[1][5].quality = input_1[5].quality /\
list_to_reorder[1][6].length = input_1[6].length /\
list_to_reorder[1][6].quality = input_1[6].quality /\
swapping_decisions[1][1] = input_2[1] /\
swapping_decisions[1][2] = input_2[2] /\
swapping_decisions[1][3] = input_2[3] /\
swapping_decisions[1][4] = input_2[4] /\
swapping_decisions[1][5] = input_2[5] /\
new_list[1][1].length = list_to_reorder[1][1].length /\
new_list[1][1].quality = list_to_reorder[1][1].quality /\
new_list[1][2].length = list_to_reorder[1][2].length /\
new_list[1][2].quality = list_to_reorder[1][2].quality /\
new_list[1][3].length = list_to_reorder[1][3].length /\
new_list[1][3].quality = list_to_reorder[1][3].quality /\

```

```

new_list[1][4].length = list_to_reorder[1][4].length /\
new_list[1][4].quality = list_to_reorder[1][4].quality /\
new_list[1][5].length = list_to_reorder[1][5].length /\
new_list[1][5].quality = list_to_reorder[1][5].quality /\
new_list[1][6].length = list_to_reorder[1][6].length /\
new_list[1][6].quality = list_to_reorder[1][6].quality /\
(swapping_decisions[1][1] -> aux_piece[1].length = new_list
[1][1].length) /\
(swapping_decisions[1][1] -> aux_piece[1].quality = new_list
[1][1].quality) /\
(swapping_decisions[1][1] -> new_list[2][2].length = new_list
[1][2].length) /\
(swapping_decisions[1][1] -> new_list[2][2].quality = new_list
[1][2].quality) /\
(swapping_decisions[1][1] -> new_list[2][3].length = new_list
[1][3].length) /\
(swapping_decisions[1][1] -> new_list[2][3].quality = new_list
[1][3].quality) /\
(swapping_decisions[1][1] -> new_list[2][4].length = new_list
[1][4].length) /\
(swapping_decisions[1][1] -> new_list[2][4].quality = new_list
[1][4].quality) /\
(swapping_decisions[1][1] -> new_list[2][5].length = new_list
[1][5].length) /\
(swapping_decisions[1][1] -> new_list[2][5].quality = new_list
[1][5].quality) /\
(swapping_decisions[1][1] -> new_list[2][6].length = new_list
[1][6].length) /\
(swapping_decisions[1][1] -> new_list[2][6].quality = new_list
[1][6].quality) /\
(swapping_decisions[1][1] -> new_list[2][1].length = new_list
[1][(1 + 1)].length) /\
(swapping_decisions[1][1] -> new_list[2][1].quality = new_list
[1][(1 + 1)].quality) /\
(swapping_decisions[1][1] -> new_list[3][1].length = new_list
[2][1].length) /\
(swapping_decisions[1][1] -> new_list[3][1].quality = new_list
[2][1].quality) /\
(swapping_decisions[1][1] -> new_list[3][2].length = new_list
[2][2].length) /\
(swapping_decisions[1][1] -> new_list[3][2].quality = new_list
[2][2].quality) /\
(swapping_decisions[1][1] -> new_list[3][3].length = new_list
[2][3].length) /\
(swapping_decisions[1][1] -> new_list[3][3].quality = new_list
[2][3].quality) /\
(swapping_decisions[1][1] -> new_list[3][4].length = new_list
[2][4].length) /\
(swapping_decisions[1][1] -> new_list[3][4].quality = new_list

```

```

[2][4].quality) /\
(swapping_decisions[1][1] -> new_list[3][5].length = new_list
[2][5].length) /\
(swapping_decisions[1][1] -> new_list[3][5].quality = new_list
[2][5].quality) /\
(swapping_decisions[1][1] -> new_list[3][6].length = new_list
[2][6].length) /\
(swapping_decisions[1][1] -> new_list[3][6].quality = new_list
[2][6].quality) /\
(swapping_decisions[1][1] -> new_list[3][(1 + 1)].length =
aux_piece[1].length) /\
(swapping_decisions[1][1] -> new_list[3][(1 + 1)].quality =
aux_piece[1].quality) /\
((not swapping_decisions[1][1]) -> new_list[3] = new_list[1]) /\
(swapping_decisions[1][2] -> aux_piece[2].length = new_list
[3][2].length) /\
(swapping_decisions[1][2] -> aux_piece[2].quality = new_list
[3][2].quality) /\
(swapping_decisions[1][2] -> new_list[4][1].quality = new_list
[3][1].quality) /\
(swapping_decisions[1][2] -> new_list[4][1].length = new_list
[3][1].length) /\
(swapping_decisions[1][2] -> new_list[4][3].quality = new_list
[3][3].quality) /\
(swapping_decisions[1][2] -> new_list[4][3].length = new_list
[3][3].length) /\
(swapping_decisions[1][2] -> new_list[4][4].quality = new_list
[3][4].quality) /\
(swapping_decisions[1][2] -> new_list[4][4].length = new_list
[3][4].length) /\
(swapping_decisions[1][2] -> new_list[4][5].quality = new_list
[3][5].quality) /\
(swapping_decisions[1][2] -> new_list[4][5].length = new_list
[3][5].length) /\
(swapping_decisions[1][2] -> new_list[4][6].quality = new_list
[3][6].quality) /\
(swapping_decisions[1][2] -> new_list[4][6].length = new_list
[3][6].length) /\
(swapping_decisions[1][2] -> new_list[4][2].length = new_list
[3][(2 + 1)].length) /\
(swapping_decisions[1][2] -> new_list[4][2].quality = new_list
[3][(2 + 1)].quality) /\
(swapping_decisions[1][2] -> new_list[5][1].quality = new_list
[4][1].quality) /\
(swapping_decisions[1][2] -> new_list[5][1].length = new_list
[4][1].length) /\
(swapping_decisions[1][2] -> new_list[5][2].quality = new_list
[4][2].quality) /\
(swapping_decisions[1][2] -> new_list[5][2].length = new_list

```

```

[4][2].length) /\
(swapping_decisions[1][2] -> new_list[5][3].quality = new_list
[4][3].quality) /\
(swapping_decisions[1][2] -> new_list[5][3].length = new_list
[4][3].length) /\
(swapping_decisions[1][2] -> new_list[5][4].quality = new_list
[4][4].quality) /\
(swapping_decisions[1][2] -> new_list[5][4].length = new_list
[4][4].length) /\
(swapping_decisions[1][2] -> new_list[5][5].quality = new_list
[4][5].quality) /\
(swapping_decisions[1][2] -> new_list[5][5].length = new_list
[4][5].length) /\
(swapping_decisions[1][2] -> new_list[5][6].quality = new_list
[4][6].quality) /\
(swapping_decisions[1][2] -> new_list[5][6].length = new_list
[4][6].length) /\
(swapping_decisions[1][2] -> new_list[5][(2 + 1)].length =
aux_piece[2].length) /\
(swapping_decisions[1][2] -> new_list[5][(2 + 1)].quality =
aux_piece[2].quality) /\
((not swapping_decisions[1][2]) -> aux_piece[2] = aux_piece[1])
/\
((not swapping_decisions[1][2]) -> new_list[5] = new_list[3]) /\
(swapping_decisions[1][3] -> aux_piece[3].length = new_list
[5][3].length) /\
(swapping_decisions[1][3] -> aux_piece[3].quality = new_list
[5][3].quality) /\
(swapping_decisions[1][3] -> new_list[6][1].quality = new_list
[5][1].quality) /\
(swapping_decisions[1][3] -> new_list[6][1].length = new_list
[5][1].length) /\
(swapping_decisions[1][3] -> new_list[6][2].quality = new_list
[5][2].quality) /\
(swapping_decisions[1][3] -> new_list[6][2].length = new_list
[5][2].length) /\
(swapping_decisions[1][3] -> new_list[6][4].quality = new_list
[5][4].quality) /\
(swapping_decisions[1][3] -> new_list[6][4].length = new_list
[5][4].length) /\
(swapping_decisions[1][3] -> new_list[6][5].quality = new_list
[5][5].quality) /\
(swapping_decisions[1][3] -> new_list[6][5].length = new_list
[5][5].length) /\
(swapping_decisions[1][3] -> new_list[6][6].quality = new_list
[5][6].quality) /\
(swapping_decisions[1][3] -> new_list[6][6].length = new_list
[5][6].length) /\
(swapping_decisions[1][3] -> new_list[6][3].length = new_list

```

```

[5][(3 + 1)].length) /\
(swapping_decisions[1][3] -> new_list[6][3].quality = new_list
[5][(3 + 1)].quality) /\
(swapping_decisions[1][3] -> new_list[7][1].quality = new_list
[6][1].quality) /\
(swapping_decisions[1][3] -> new_list[7][1].length = new_list
[6][1].length) /\
(swapping_decisions[1][3] -> new_list[7][2].quality = new_list
[6][2].quality) /\
(swapping_decisions[1][3] -> new_list[7][2].length = new_list
[6][2].length) /\
(swapping_decisions[1][3] -> new_list[7][3].quality = new_list
[6][3].quality) /\
(swapping_decisions[1][3] -> new_list[7][3].length = new_list
[6][3].length) /\
(swapping_decisions[1][3] -> new_list[7][4].quality = new_list
[6][4].quality) /\
(swapping_decisions[1][3] -> new_list[7][4].length = new_list
[6][4].length) /\
(swapping_decisions[1][3] -> new_list[7][5].quality = new_list
[6][5].quality) /\
(swapping_decisions[1][3] -> new_list[7][5].length = new_list
[6][5].length) /\
(swapping_decisions[1][3] -> new_list[7][6].quality = new_list
[6][6].quality) /\
(swapping_decisions[1][3] -> new_list[7][6].length = new_list
[6][6].length) /\
(swapping_decisions[1][3] -> new_list[7][(3 + 1)].length =
aux_piece[3].length) /\
(swapping_decisions[1][3] -> new_list[7][(3 + 1)].quality =
aux_piece[3].quality) /\
((not swapping_decisions[1][3]) -> aux_piece[3] = aux_piece[2])
/\
((not swapping_decisions[1][3]) -> new_list[7] = new_list[5]) /\
(swapping_decisions[1][4] -> aux_piece[4].length = new_list
[7][4].length) /\
(swapping_decisions[1][4] -> aux_piece[4].quality = new_list
[7][4].quality) /\
(swapping_decisions[1][4] -> new_list[8][1].quality = new_list
[7][1].quality) /\
(swapping_decisions[1][4] -> new_list[8][1].length = new_list
[7][1].length) /\
(swapping_decisions[1][4] -> new_list[8][2].quality = new_list
[7][2].quality) /\
(swapping_decisions[1][4] -> new_list[8][2].length = new_list
[7][2].length) /\
(swapping_decisions[1][4] -> new_list[8][3].quality = new_list
[7][3].quality) /\
(swapping_decisions[1][4] -> new_list[8][3].length = new_list

```

```

    [7][3].length) /\
    (swapping_decisions[1][4] -> new_list[8][5].quality = new_list
    [7][5].quality) /\
    (swapping_decisions[1][4] -> new_list[8][5].length = new_list
    [7][5].length) /\
    (swapping_decisions[1][4] -> new_list[8][6].quality = new_list
    [7][6].quality) /\
    (swapping_decisions[1][4] -> new_list[8][6].length = new_list
    [7][6].length) /\
    (swapping_decisions[1][4] -> new_list[8][4].length = new_list
    [7][(4 + 1)].length) /\
    (swapping_decisions[1][4] -> new_list[8][4].quality = new_list
    [7][(4 + 1)].quality) /\
    (swapping_decisions[1][4] -> new_list[9][1].quality = new_list
    [8][1].quality) /\
    (swapping_decisions[1][4] -> new_list[9][1].length = new_list
    [8][1].length) /\
    (swapping_decisions[1][4] -> new_list[9][2].quality = new_list
    [8][2].quality) /\
    (swapping_decisions[1][4] -> new_list[9][2].length = new_list
    [8][2].length) /\
    (swapping_decisions[1][4] -> new_list[9][3].quality = new_list
    [8][3].quality) /\
    (swapping_decisions[1][4] -> new_list[9][3].length = new_list
    [8][3].length) /\
    (swapping_decisions[1][4] -> new_list[9][4].quality = new_list
    [8][4].quality) /\
    (swapping_decisions[1][4] -> new_list[9][4].length = new_list
    [8][4].length) /\
    (swapping_decisions[1][4] -> new_list[9][5].quality = new_list
    [8][5].quality) /\
    (swapping_decisions[1][4] -> new_list[9][5].length = new_list
    [8][5].length) /\
    (swapping_decisions[1][4] -> new_list[9][6].quality = new_list
    [8][6].quality) /\
    (swapping_decisions[1][4] -> new_list[9][6].length = new_list
    [8][6].length) /\
    (swapping_decisions[1][4] -> new_list[9][(4 + 1)].length =
    aux_piece[4].length) /\
    (swapping_decisions[1][4] -> new_list[9][(4 + 1)].quality =
    aux_piece[4].quality) /\
    ((not swapping_decisions[1][4]) -> aux_piece[4] = aux_piece[3])
    /\
    ((not swapping_decisions[1][4]) -> new_list[9] = new_list[7]) /\
    output_1 = new_list[9]
);
int: N_BOARDS = 2;
int: MAX_BOARD_LENGTH = 30;
int: MAX_N_INTERVALS = 2;

```

```

int: MAX_N_CUTS_PER_BOARD = 4;
int: N_PIECES = 6;
int: BEAM_LENGTH = 10;
int: BEAM_DEPTH = 3;
int: MAX_PIECES_PER_BEAM = 2;
int: MIN_DIST_BETWEEN_PIECES = 1;
array[1..2] of array[1..2] of int: FORBIDDEN_INTERVALS = [[3, 4], [7,
8]];
array[1..2] of Board: GIVEN_INITIAL_BOARDS = [(
    length: 20,
    bad_intervals: [(
        start: 5,
        end: 6
    ), (
        start: 15,
        end: 16
    )],
    curved_intervals: [(
        start: 10,
        end: 12
    ), (
        start: 18,
        end: 20
    )]
), (
    length: 25,
    bad_intervals: [(
        start: 10,
        end: 14
    ), (
        start: 18,
        end: 20
    )],
    curved_intervals: [(
        start: 7,
        end: 9
    ), (
        start: 18,
        end: 20
    )]
)];
array[1..1] of array[1..2] of record(
    var 0..MAX_BOARD_LENGTH: length,
    array[1..2] of record(
        var 0..MAX_BOARD_LENGTH: start,
        var 0..MAX_BOARD_LENGTH: end
    ): bad_intervals,
    array[1..2] of record(
        var 0..MAX_BOARD_LENGTH: start,

```

```

    var 0..MAX_BOARD_LENGTH: end
): curved_intervals
): initial_boards;
array[1..1] of array[1..2] of record(
    array[1..4] of var 0..MAX_BOARD_LENGTH: position_list
): cuts_list_list;
array[1..1] of array[1..6] of record(
    var 0..MAX_BOARD_LENGTH: length,
    var 0..1: quality
): pieces;
array[1..6] of array[1..2] of record(
    var 0..MAX_BOARD_LENGTH: start,
    var 0..MAX_BOARD_LENGTH: end
): bad_intervals_board__cutting_machine__1;
array[1..1] of array[1..2] of record(
    var 0..MAX_BOARD_LENGTH: length,
    array[1..2] of record(
        var 0..MAX_BOARD_LENGTH: start,
        var 0..MAX_BOARD_LENGTH: end
    )
): bad_intervals,
    array[1..2] of record(
        var 0..MAX_BOARD_LENGTH: start,
        var 0..MAX_BOARD_LENGTH: end
    )
): curved_intervals
): board_list__cutting_machine__1;
array[1..2] of array[1..2] of record(
    var 0..MAX_BOARD_LENGTH: start,
    var 0..MAX_BOARD_LENGTH: end
): curved_intervals_board__cutting_machine__1;
array[1..2] of array[1..4] of var 0..MAX_BOARD_LENGTH:
    cut_list__cutting_machine__1;
array[1..1] of array[1..2] of record(
    array[1..4] of var 0..MAX_BOARD_LENGTH: position_list
): cuts_list_list__cutting_machine__1;
array[1..6] of var int: piece_length__cutting_machine__1;
array[1..7] of array[1..6] of record(
    var 0..MAX_BOARD_LENGTH: length,
    var 0..1: quality
): pieces__cutting_machine__1;
array[1..6] of var bool: quality__cutting_machine__1;
array[1..1] of array[1..6] of var bool: keep_decisions;
array[1..1] of array[1..6] of record(
    var 0..MAX_BOARD_LENGTH: length,
    var 0..1: quality
): filtered_pieces;
array[1..1] of var int: waste;
array[1..1] of array[1..6] of record(
    var 0..MAX_BOARD_LENGTH: length,
    var 0..1: quality

```

```

): filtered_list__filtering_machine__1;
array[1..1] of array[1..6] of var bool:
    keep_decisions__filtering_machine__1;
array[1..1] of array[1..6] of record(
    var 0..MAX_BOARD_LENGTH: length,
    var 0..1: quality
): list_to_filter__filtering_machine__1;
array[1..6] of var int: waste__filtering_machine__1;
array[1..1] of array[1..5] of var bool: swapping_decisions;
array[1..1] of array[1..6] of record(
    var 0..MAX_BOARD_LENGTH: length,
    var 0..1: quality
): reordered_pieces;
array[1..4] of record(
    var 0..MAX_BOARD_LENGTH: length,
    var 0..1: quality
): aux_piece__reordering_piece_machine__1;
array[1..1] of array[1..6] of record(
    var 0..MAX_BOARD_LENGTH: length,
    var 0..1: quality
): list_to_reorder__reordering_piece_machine__1;
array[1..9] of array[1..6] of record(
    var 0..MAX_BOARD_LENGTH: length,
    var 0..1: quality
): new_list__reordering_piece_machine__1;
array[1..1] of array[1..5] of var bool:
    swapping_decisions__reordering_piece_machine__1;
array[1..7] of var int: current_index__checking_machine__1;
array[1..13] of var int: depth__checking_machine__1;
array[1..1] of var int: end__checking_machine__1;
array[1..13] of var int: length__checking_machine__1;
array[1..13] of var int: n_length__checking_machine__1;
array[1..7] of var int: n_prev_layer__checking_machine__1;
array[1..7] of var int: new_beam__checking_machine__1;
array[1..1] of array[1..6] of record(
    var 0..MAX_BOARD_LENGTH: length,
    var 0..1: quality
): pieces__checking_machine__1;
array[1..7] of var int: s__checking_machine__1;
array[1..1] of var int: start__checking_machine__1;
constraint initial_boards[1][1].length = GIVEN_INITIAL_BOARDS[1].
    length;
constraint initial_boards[1][1].bad_intervals[1].start =
    GIVEN_INITIAL_BOARDS[1].bad_intervals[1].start;
constraint initial_boards[1][1].bad_intervals[1].end =
    GIVEN_INITIAL_BOARDS[1].bad_intervals[1].end;
constraint initial_boards[1][1].bad_intervals[2].start =
    GIVEN_INITIAL_BOARDS[1].bad_intervals[2].start;
constraint initial_boards[1][1].bad_intervals[2].end =

```

```

    GIVEN_INITIAL_BOARDS[1].bad_intervals[2].end;
constraint initial_boards[1][1].curved_intervals[1].start =
    GIVEN_INITIAL_BOARDS[1].curved_intervals[1].start;
constraint initial_boards[1][1].curved_intervals[1].end =
    GIVEN_INITIAL_BOARDS[1].curved_intervals[1].end;
constraint initial_boards[1][1].curved_intervals[2].start =
    GIVEN_INITIAL_BOARDS[1].curved_intervals[2].start;
constraint initial_boards[1][1].curved_intervals[2].end =
    GIVEN_INITIAL_BOARDS[1].curved_intervals[2].end;
constraint initial_boards[1][2].length = GIVEN_INITIAL_BOARDS[2].
    length;
constraint initial_boards[1][2].bad_intervals[1].start =
    GIVEN_INITIAL_BOARDS[2].bad_intervals[1].start;
constraint initial_boards[1][2].bad_intervals[1].end =
    GIVEN_INITIAL_BOARDS[2].bad_intervals[1].end;
constraint initial_boards[1][2].bad_intervals[2].start =
    GIVEN_INITIAL_BOARDS[2].bad_intervals[2].start;
constraint initial_boards[1][2].bad_intervals[2].end =
    GIVEN_INITIAL_BOARDS[2].bad_intervals[2].end;
constraint initial_boards[1][2].curved_intervals[1].start =
    GIVEN_INITIAL_BOARDS[2].curved_intervals[1].start;
constraint initial_boards[1][2].curved_intervals[1].end =
    GIVEN_INITIAL_BOARDS[2].curved_intervals[1].end;
constraint initial_boards[1][2].curved_intervals[2].start =
    GIVEN_INITIAL_BOARDS[2].curved_intervals[2].start;
constraint initial_boards[1][2].curved_intervals[2].end =
    GIVEN_INITIAL_BOARDS[2].curved_intervals[2].end;
constraint cutting_machine(initial_boards[1], cuts_list_list[1],
    pieces[1], bad_intervals_board__cutting_machine__1,
    board_list__cutting_machine__1,
    curved_intervals_board__cutting_machine__1,
    cut_list__cutting_machine__1, cuts_list_list__cutting_machine__1,
    piece_length__cutting_machine__1, pieces__cutting_machine__1,
    quality__cutting_machine__1);
constraint filtering_machine(pieces[1], keep_decisions[1],
    filtered_pieces[1], waste[1], filtered_list__filtering_machine__1,
    keep_decisions__filtering_machine__1,
    list_to_filter__filtering_machine__1, waste__filtering_machine__1)
;
constraint reordering_piece_machine(filtered_pieces[1],
    swapping_decisions[1], reordered_pieces[1],
    aux_piece__reordering_piece_machine__1,
    list_to_reorder__reordering_piece_machine__1,
    new_list__reordering_piece_machine__1,
    swapping_decisions__reordering_piece_machine__1);
constraint checking_machine(reordered_pieces[1],
    current_index__checking_machine__1, depth__checking_machine__1,
    end__checking_machine__1, length__checking_machine__1,
    n_length__checking_machine__1, n_prev_layer__checking_machine__1,

```

```
new_beam__checking_machine__1, pieces__checking_machine__1,  
s__checking_machine__1, start__checking_machine__1);  
solve minimize waste[1];
```