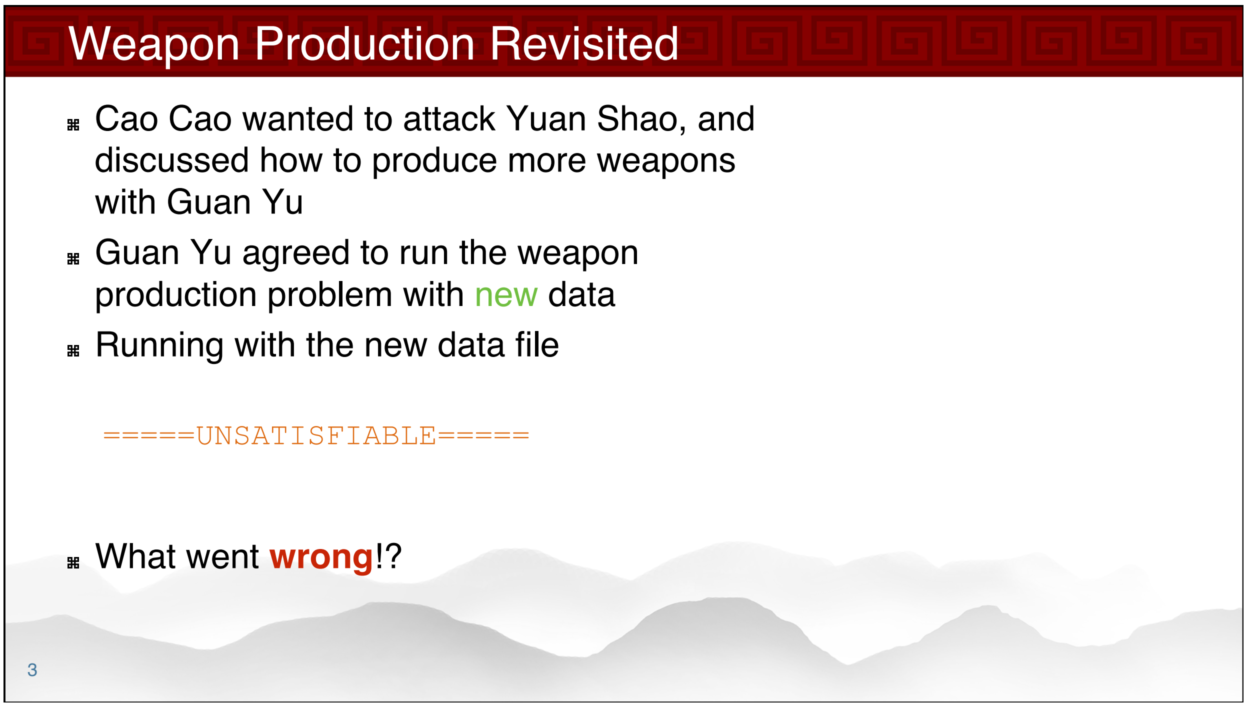
Model Debugging

Jimmy Lee & Peter Stuckey

Weapon Production Revisited



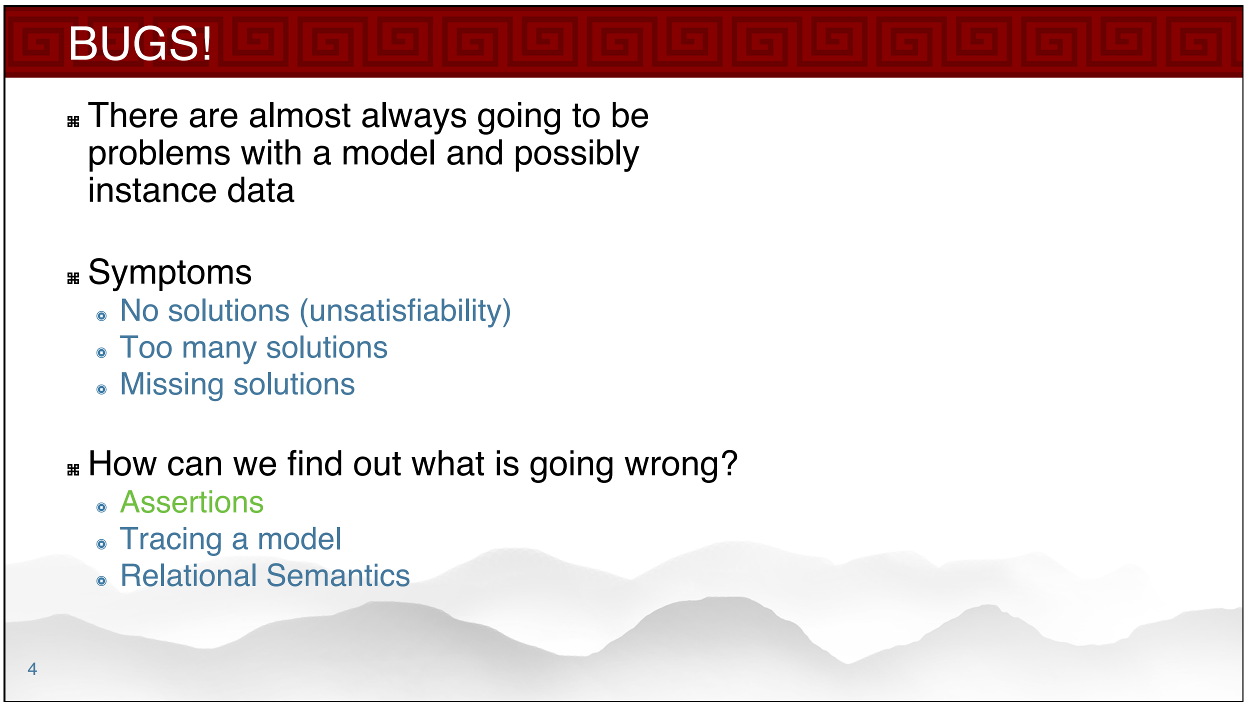
Weapon Production Revisited

- Cao Cao wanted to attack Yuan Shao, and discussed how to produce more weapons with Guan Yu
- Guan Yu agreed to run the weapon production problem with new data
- Running with the new data file

```
=====UNSATISFIABLE=====
```

- What went **wrong**!?

3



BUGS!

- There are almost always going to be problems with a model and possibly instance data
- Symptoms
  - No solutions (unsatisfiability)
  - Too many solutions
  - Missing solutions
- How can we find out what is going wrong?
  - Assertions
  - Tracing a model
  - Relational Semantics

4

Assertions

- Defensive programming requires that we check that the data values are valid
  - `assert(`*boolexp*,*stringexp*`)`
    - returns true if *boolexp* holds,
    - otherwise prints *stringexp* and aborts
- For example

```
array[RESOURCE] of int: capacity;
constraint assert(
    forall(r in RESOURCE)(capacity[r] >= 0),
    "Error: negative capacity");
```

Assertions Exercise

- Write an assertion to check that consumption is non-negative where

```
array[PRODUCT,RESOURCE] of int: consumption;

constraint forall(p in PRODUCT,
                  r in RESOURCE)
   (assert(consumption[p,r] >= 0,
           "consumption[\(p),\(r)] < 0!"));
```

- A better error message than the previous example

Solving the Model

- Insert assertion for parameters profit, capacity and consumption
- Run again …

```
MiniZinc: evaluation error:
prod-plan.mzn:12:
 in call 'forall'
     with r = 3
prod-plan.mzn:13:
 in call 'assert'
   Assertion failed: capacity[3] < 0!
```

- It turns out that the third piece of data for the capacity array was **entered by mistake** as a **-ve** quantity in the data file

Looking at the Data

- Here is what was entered

```
PRODUCT = {PIKE, SWORD, AXE, SPEAR, CLUB};
profit =  [20.0, 18.0, 3.0, 4.0, 2.0];
RESOURCE = {IRON, WOOD, SMITH, CARPENTER};
capacity = [5000, 7500, -2000, 10000];
consumption = [| 1.5, 1.0, 0.5, 2.0
               | 1.0, 0.0, 1.0, 0.0
               | 0.0, 1.0, 0.0, 1.0
               | 0.1, 1.0, 0.0, 1.0
               | 0.0, 0.5, 0.0, 1.0 |];
```

- In general erroneous data can be hard to spot

Assertions again

- There is another form of assertion with three arguments:
  - `assert`(*boolexp*,*stringexp*,*exp*)
    - returns *exp* if *boolexp* holds,
    - otherwise prints *stringexp* and aborts
- Useful when not all of a model will be executed, in particular later when we introduce predicates and user-defined functions

9

Assertions for Debugging

- You can (ab)use assertions to help debug

```
int: n = 5;
array[1..n] of var 1..n: a;
array[1..n] of 1..n: b = [3,5,2,3,1];

constraint forall(j in 1..n, i in b[n-j]..n)
   (a[j] < i);
```

- Error message

```
MiniZinc: result of evaluation is undefined:
debug1.mzn:5:
  in call 'forall'
  in array comprehension expression
    with j = 5
  in binary '..' operator expression
  in array access
  array access out of bounds
```

10

Assertions for Debugging

- You can (ab)use assertions to help debug

```
int: n = 5;
 array[1..n] of var 1..n: a;
 array[1..n] of 1..n: b = [3,5,2,3,1];

 constraint forall(j in 1..n)
    (assert(n-j in 1..n, "b[\(n-j)]"));
```

- Error message

```
debug.mzn:5
  In constraint.
  In 'forall' expression.
  In comprehension.
  j = 5
  In comprehension head.
  In assert expression
  Assertion failure: "b[0]"
```

11

Out of Range Errors in Constraints

- You can (ab)use assertions to help debug

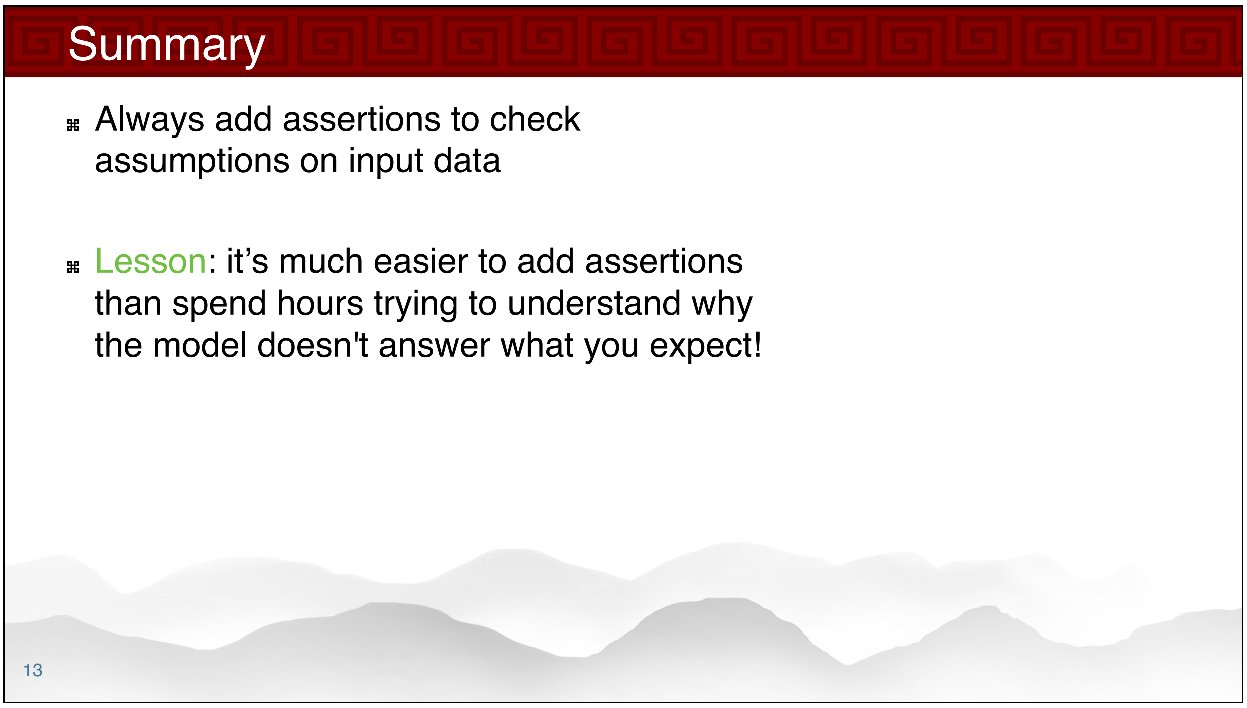
```
int: n = 5;
 array[1..n] of var 1..n: a;
 array[1..n] of 1..n: b = [3,5,2,3,1];

 constraint forall(j in 1..n)(a[n-j] < b[j]);
```

- Error message

```
debug.mzn:5
  In constraint.
  In 'forall' expression.
  Model inconsistency detected
```

- What's going on?
  - `a[0] < b[5]` ?????????
  - relational semantics

12



Summary

- Always add assertions to check assumptions on input data
- Lesson: it's much easier to add assertions than spend hours trying to understand why the model doesn't answer what you expect!

13

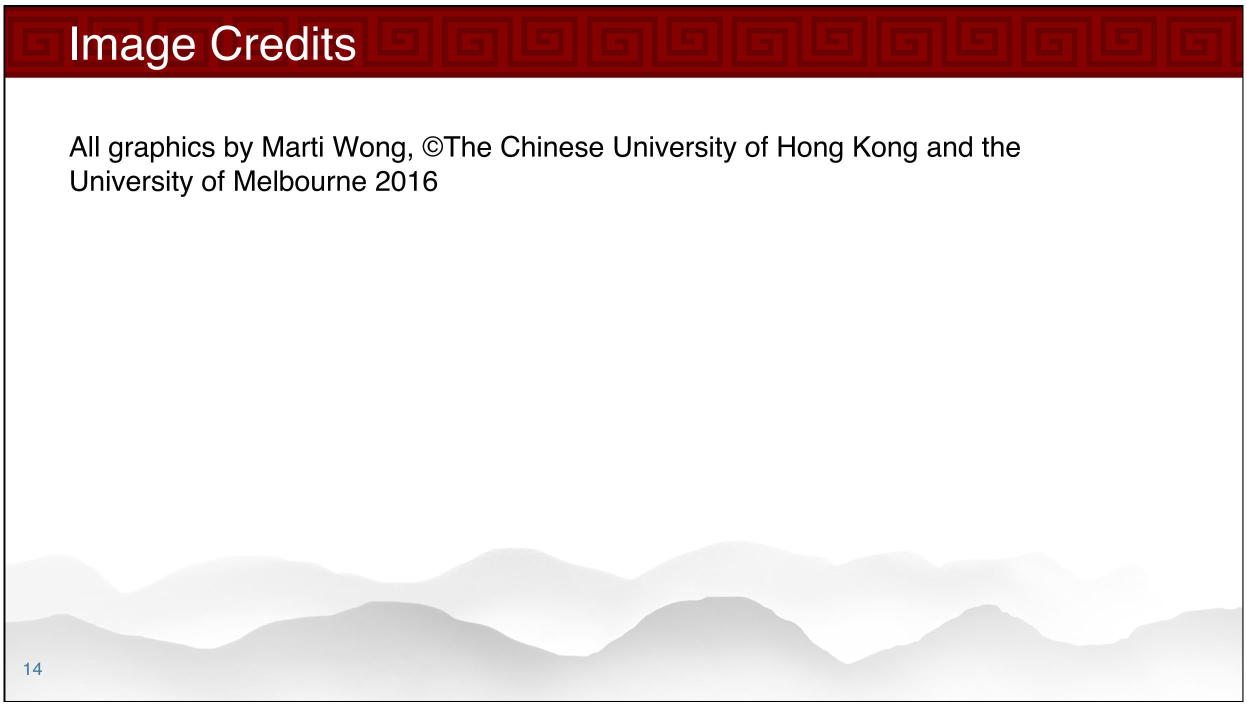


Image Credits

All graphics by Marti Wong, ©The Chinese University of Hong Kong and the University of Melbourne 2016

14