THE UNIVERSITY OF MELBOURNE 香港中文大學 The Chinese University of Hong Kong

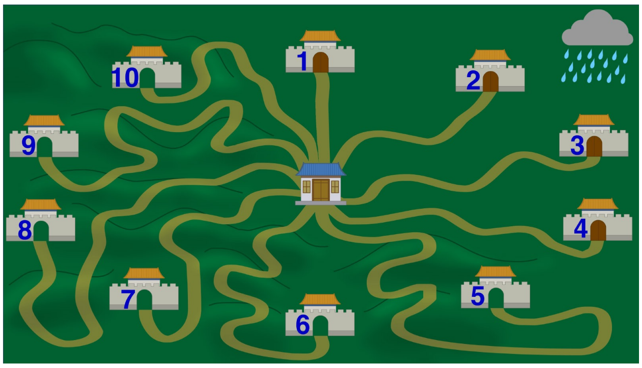
Relational Semantics

Jimmy Lee & Peter Stuckey

香港中文大學 The Chinese University of Hong Kong THE UNIVERSITY OF MELBOURNE

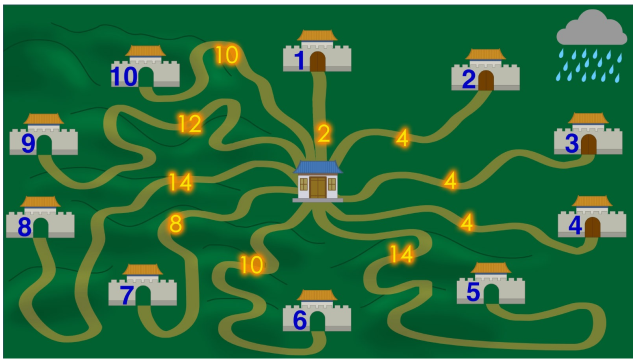
Sending out an Emissary

The Escape Routes



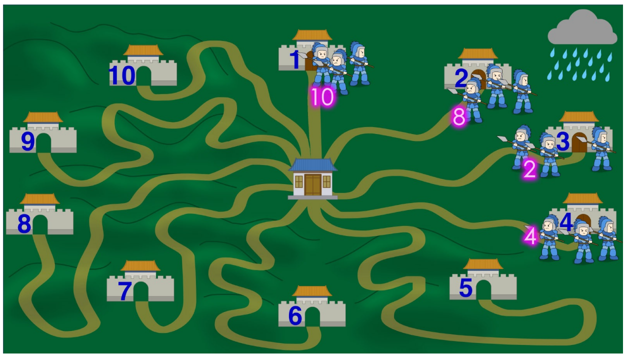
3

The Costs—Distance

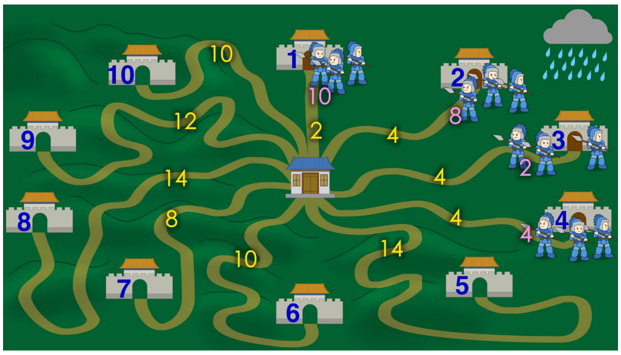


4

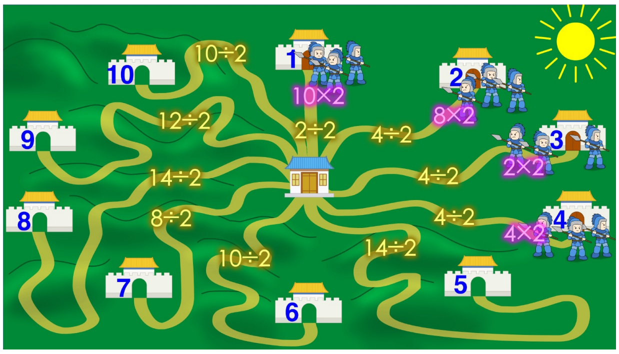
The Costs—Guards



Weather Effect—Rainy

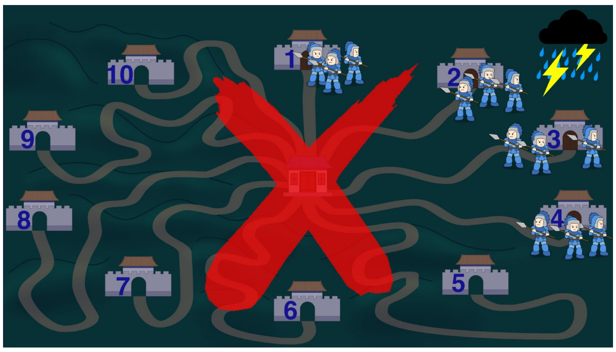


Weather Effect—Sunny



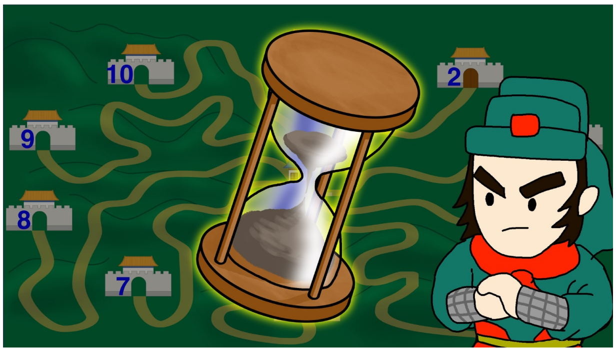
7

Weather Effect—Stormy



8

Optimizing Escape Time



9

Sending out an Emissary

- Guan Yu is under house arrest in the camp of Cao Cao. He decides to send his son Guan Ping to find news. There are 10 paths, four with a large guard post
- The time to sneak each path
  - in rain is given by an array `time`,
  - halved if sunny, impossible if stormy
- for paths with guard posts
  - extra time to avoid the guards, `guard`.
  - doubled if sunny
- Find the minimal time path

10

Sending an Emissary (sneak.mzn)

```
set of int: PATH = 1..10;
set of int: POST = 1..4;

array[PATH] of int: time;
array[POST] of int: guard;

var 0..2: weather;
  % 0 = storm, 1 = rain, 2 = sun
var PATH: path;

var int: t = time[path] div weather +
             guard[path] * weather;

solve minimize t;

time  = [ 2, 4, 4, 4, 14, 10, 8, 14, 12, 10];
guard = [10, 8, 2, 4];
```

11

Is Our Model Sensible?

- What about division by zero
  `time[path] div weather`
- What happens when weather = 0?
- x = y div z should act like
  - $y = x * z + r \wedge r < z$
- Hence z = 0 means failure
- Our model is correct
  - when weather = 0
  - Guan Ping cannot sneak out
- Hence weather = 0 will not be part of a solution

12

Sending an Emissary

- Running the model we obtain
  - `weather = 1;`
  - `path = 3;`
- Is this what we expected? `t = 6`
- What about `path = 7` ?
  `guard[path] * weather;`
- `guard[7]` is undefined ($\perp$)
- makes the entire constraint false
- in effect enforces `path in POST`

13

Partial Functions in Modeling

- Modeling makes use of partial functions
  - division
  - array access
- How should your model act when
  - division by zero
  - array access out of bounds
- The relational semantics says these are just statements with no solution

14

Fixing the Bugs (sneakFixed.mzn)

- Replace

```
var int: t = … + guard[path] * weather;
```

- With

```
var int: t = .. + extra * weather;
var int: extra;
path in POST -> extra = guard[path];
not(path in POST) -> extra = 0;
```

- Note if path = 7 then
  - `extra = guard[path]` is false
  - but so is `path in POST`, implication is true
- Or equivalently

```
var int: t = time[path] div weather +
  if path in POST then guard[path] else 0
  endif * weather;
```


Sending an Emissary

- With our corrected model we obtain
  - `weather = 2;`
  - `path = 7;`
- Waiting for sunny weather and using path 7 only requires time 4!
- Now that our model correctly uses the relational semantics!

THE UNIVERSITY OF MELBOURNE 香港中文大學 The Chinese University of Hong Kong

Relational Semantics

- MiniZinc assumes the relational semantics
- Undefined “floats” to the nearest enclosing Boolean context and becomes false
- Undefined results from partial functions
  - division by zero
  - out of bounds array lookup
- Some obvious transformations are invalid
  - `not(x div y = 1)` not same as `x div y != 1`
  - (x,y) = (0,0) is a solution of the first
- Best to write models where undefined cannot occur

17

Summary

- Relational semantics is important
  - to understand how models act
  - but most models have no partial functions!
- Remember undefined (⊥) floats to nearest enclosing Boolean context and becomes false
- But you should model to avoid partial function applications, e.g.
  - x div y assure y != 0
  - x[i] assure variable or parameter i is in the index_set of x

18

Image Credits

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