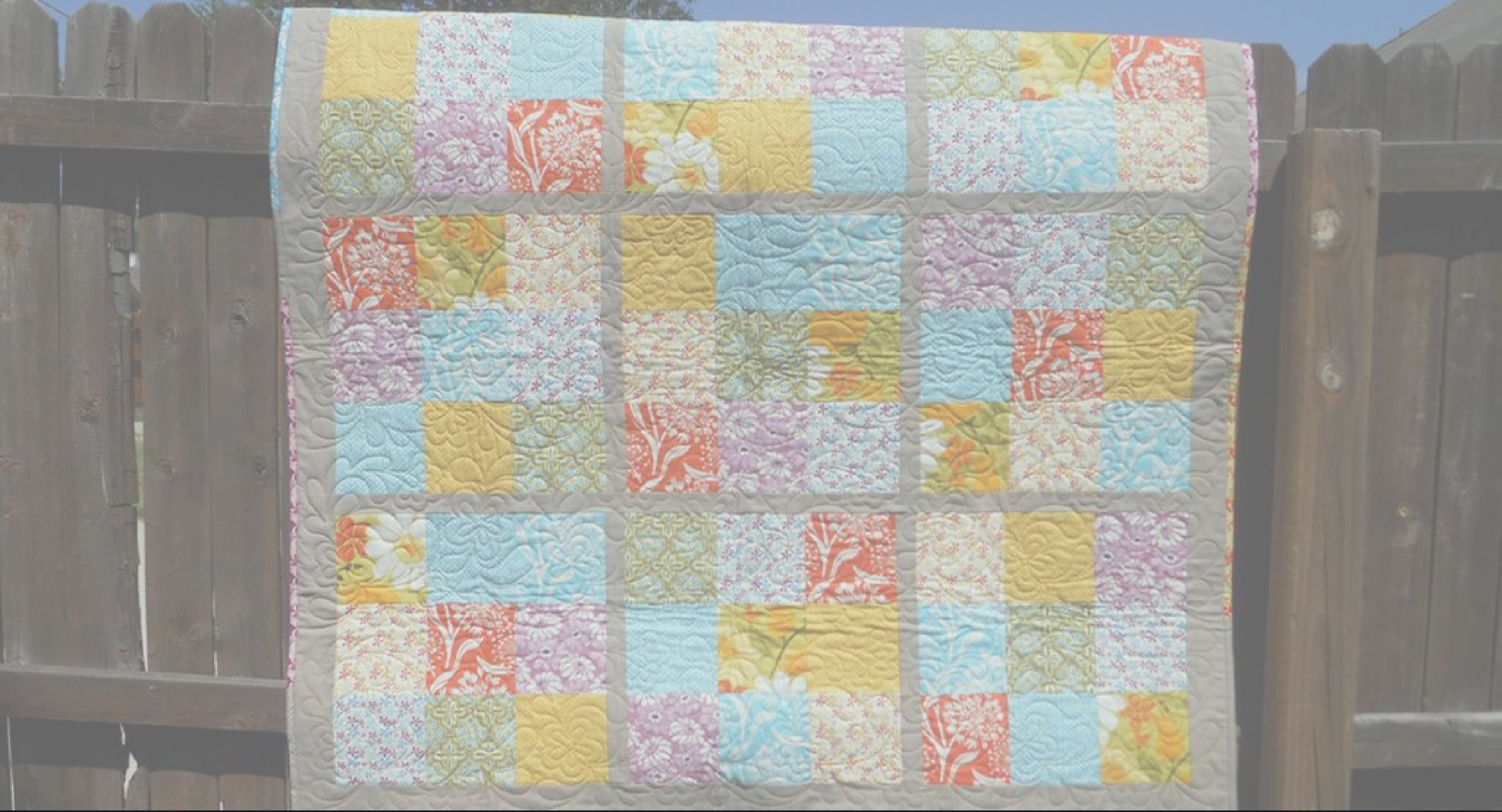


Constraint Satisfaction

constraint propagation, answer set programming

CS 4100/5100

Foundations of AI

Announcements

- Assignment 1 due today
- Assignment 2 out tomorrow
- Project pitches due next week
- Office hours next week: Tuesday 10am - noon

PROJECT PITCHES

Next week: Pitch your final project!

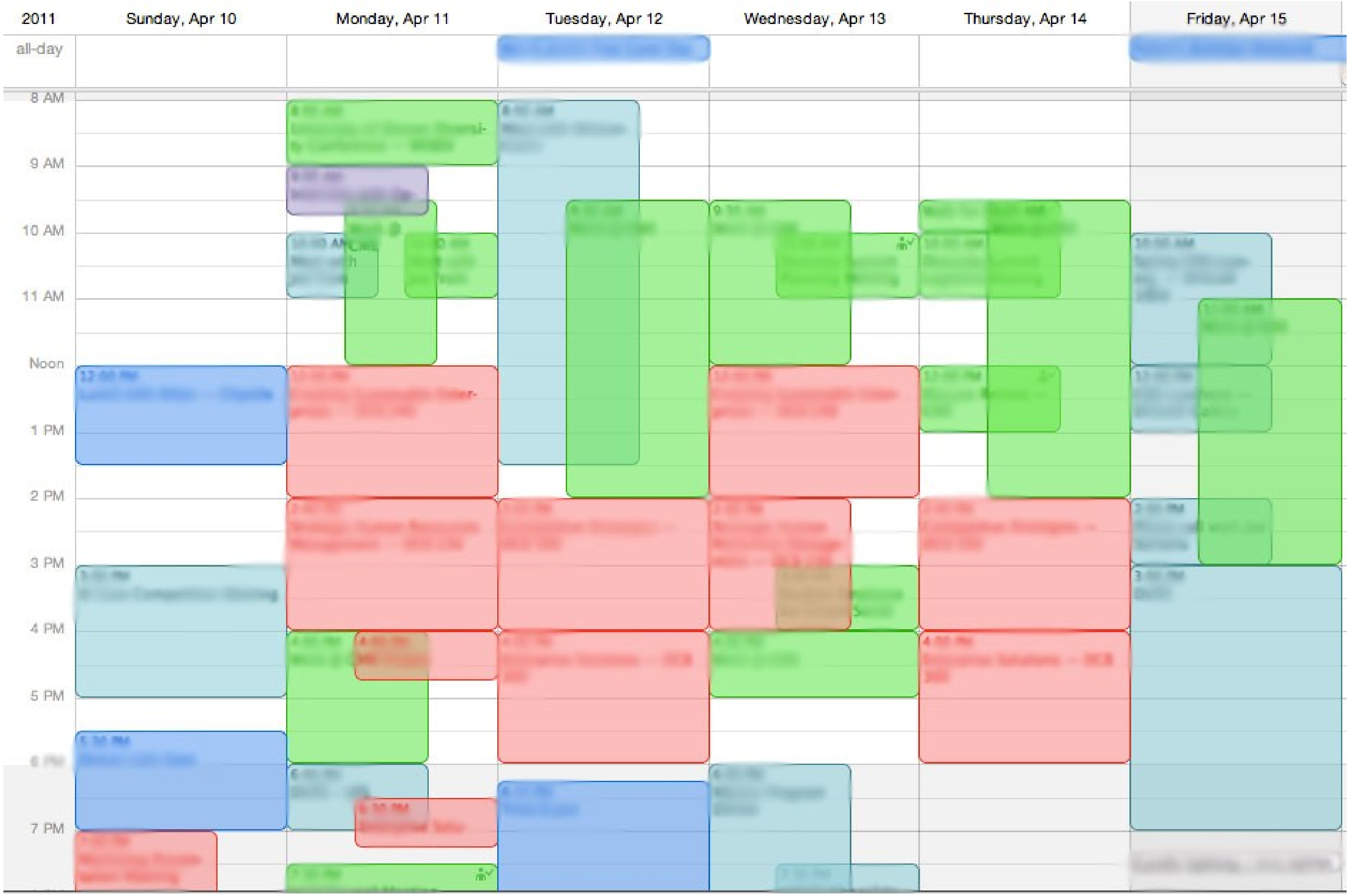
- Teams of 2-3 students
  - Exceptions: talk to me during break or end of class
- 2 minute pitch, up to 2 minutes feedback
- Application of AI to **your** interest area
- What is the problem you want to solve, or the question you want answered?
- What is your intended solution?

Potential Project Ideas

- Level generator for Super Mario World
- AI for controlling Pacman and Ghosts
- Evolutionary music or art generator
- Pattern recognition (faces? license plates?)
- Chat bot for tutoring math students

CONSTRAINT PROBLEMS

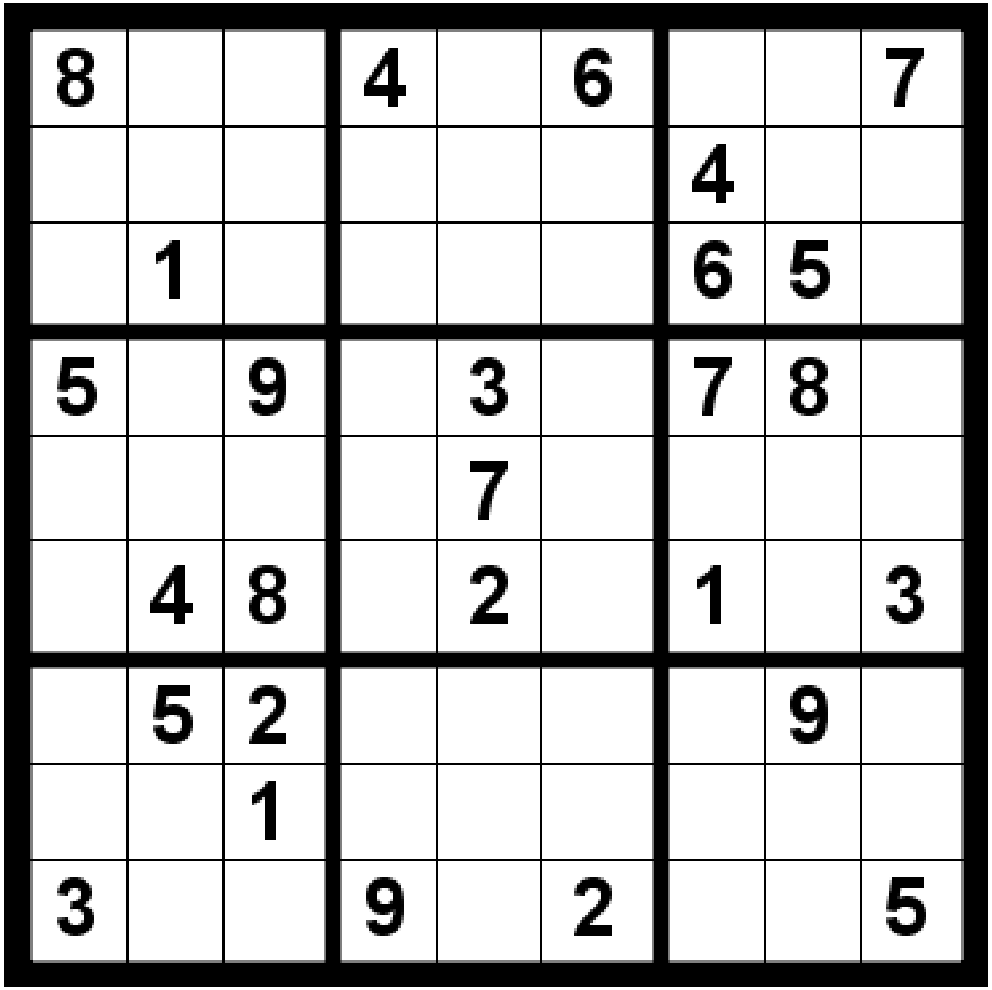
Scheduling


2011
Sunday, Apr 10
Monday, Apr 11
Tuesday, Apr 12
Wednesday, Apr 13
Thursday, Apr 14
Friday, Apr 15
all-day
8 AM
9 AM
10 AM
11 AM
Noon
1 PM
2 PM
3 PM
4 PM
5 PM
6 PM
7 PM

Scheduling

Chip Design

Puzzle Solving



8			4		6			7
						4		
	1					6	5	
5		9		3		7	8	
				7				
	4	8		2		1		3
	5	2					9	
		1						
3			9		2			5

Problem Formulation

- Variables
  - What we are solving for
- Domains
  - The values that variables can be
- Constraints
  - Allowable combinations of values

Problem Formulation: Sudoku

- Variables
  - ?
- Domains
  - ?
- Constraints
  - ?

	A	B	C	D	E	F	G	H	I
1	8			4		6			7
2							4		
3		1					6	5	
4	5		9		3		7	8	
5					7				
6		4	8		2		1		3
7		5	2					9	
8			1						
9	3			9		2			5

Problem Formulation: Sudoku

- Variables
  - All the grid squares
- Domains
  - ?
- Constraints
  - ?

	A	B	C	D	E	F	G	H	I
1	8			4		6			7
2							4		
3		1					6	5	
4	5		9		3		7	8	
5					7				
6		4	8		2		1		3
7		5	2					9	
8			1						
9	3			9		2			5

Problem Formulation: Sudoku

- Variables
  - All the grid squares
- Domains
  - A1: [1, 2, 3, 4, 5, 6, 7, 8, 9]
- Constraints
  - ?

	A	B	C	D	E	F	G	H	I
1	8			4		6			7
2							4		
3		1					6	5	
4	5		9		3		7	8	
5					7				
6		4	8		2		1		3
7		5	2					9	
8			1						
9	3			9		2			5

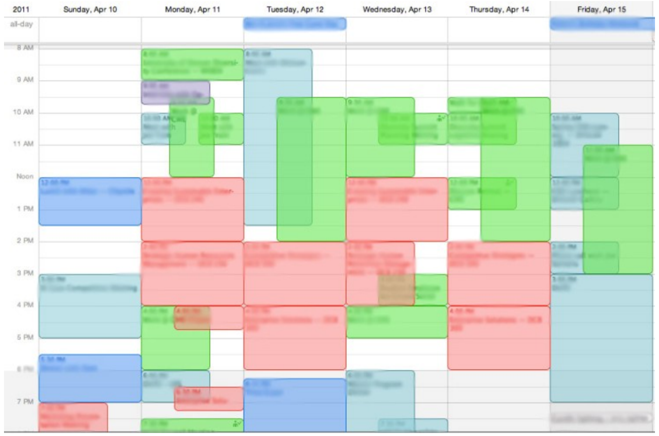
Problem Formulation: Sudoku

- Variables
  - All the grid squares
- Domains
  - A1: [1, 2, 3, 4, 5, 6, 7, 8, 9]
- Constraints
  - All different: [A1, B1, C1, A2, B2, C2, A3, B3, C3]……

	A	B	C	D	E	F	G	H	I
1	8			4		6			7
2							4		
3		1					6	5	
4	5		9		3		7	8	
5					7				
6		4	8		2		1		3
7		5	2					9	
8			1						
9	3			9		2			5

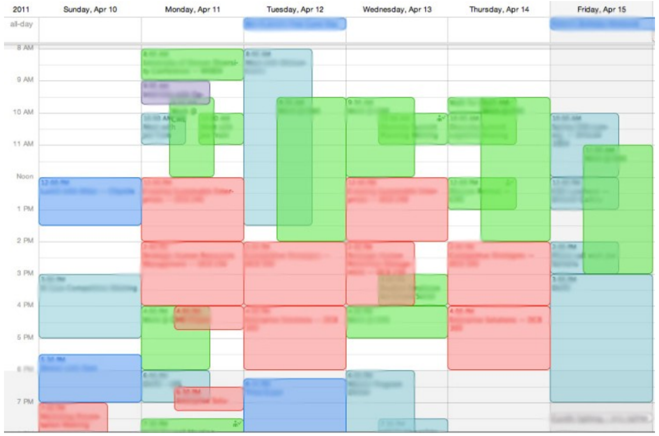
Problem Formulation: Scheduling

- Variables
  - ?
- Domains
  - ?
- Constraints
  - ?



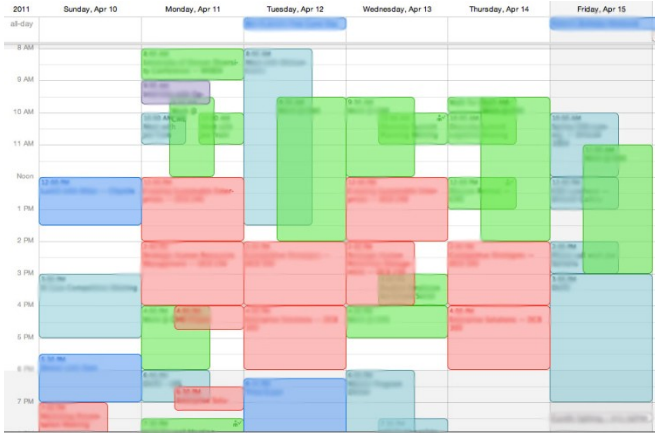
Problem Formulation: Scheduling

- Variables
  - Time slots, session lengths
- Domains
  - ?
- Constraints
  - ?



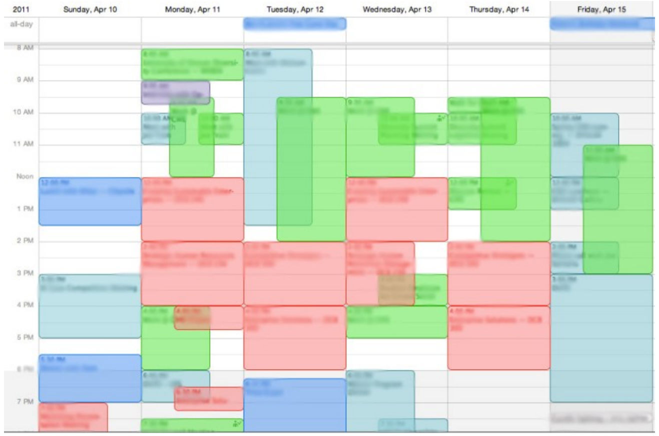
Problem Formulation: Scheduling

- Variables
  - Time slots, session lengths
- Domains
  - Time A: [calculus, AI, swim] …
- Constraints
  - ?



Problem Formulation: Scheduling

- Variables
  - Time slots, session lengths
- Domains
  - Time A: [calculus, AI, swim] …
- Constraints
  - Total time needed for a class
  - No classes on Friday
  - Co-requisite courses



Problem Formulation: Motherboard Design

- Variables
  - ?
- Domains
  - ?
- Constraints
  - ?

Problem Formulation: Motherboard Design

- Variables
  - Positions, dimensions, heat, power per component
- Domains
  - ?
- Constraints
  - ?

Problem Formulation: Motherboard Design

- Variables
  - Positions, dimensions, heat, power per component
- Domains
  - Position X: [0, 300] Y: [0, 200]
  - Power: [0, 40] watts
- Constraints
  - ?

Problem Formulation: Motherboard Design

- Variables
  - Positions, dimensions, heat, power per component
- Domains
  - Position X: [0, 300] Y: [0, 200]
  - Power: [0, 40] watts
- Constraints
  - No overlapping components
  - Heat sensitivity of nearby components

Types of Domains

- Discrete vs. continuous

- Finite vs. infinite

Types of Constraints

- Unary: single variable
- Binary: two variables
- Global: many variables
- Preference: soft requirements

Types of Constraints

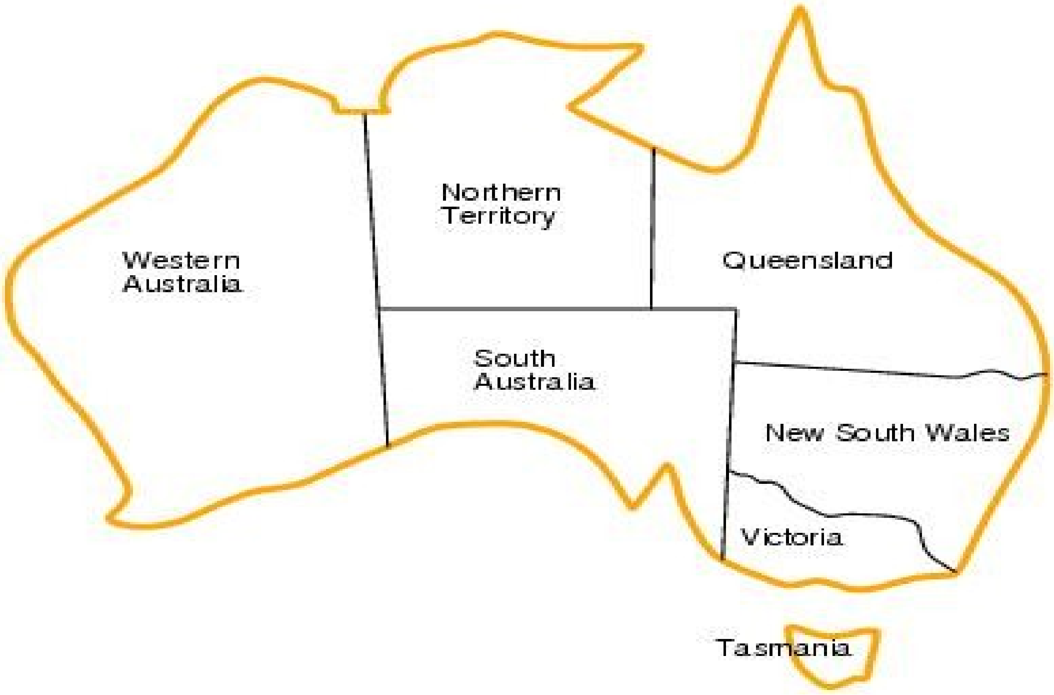
- Unary: single variable
- Binary: two variables
- Global: many variables
- Preference: soft requirements

CONSTRAINT PROPAGATION

What is Constraint Propagation?

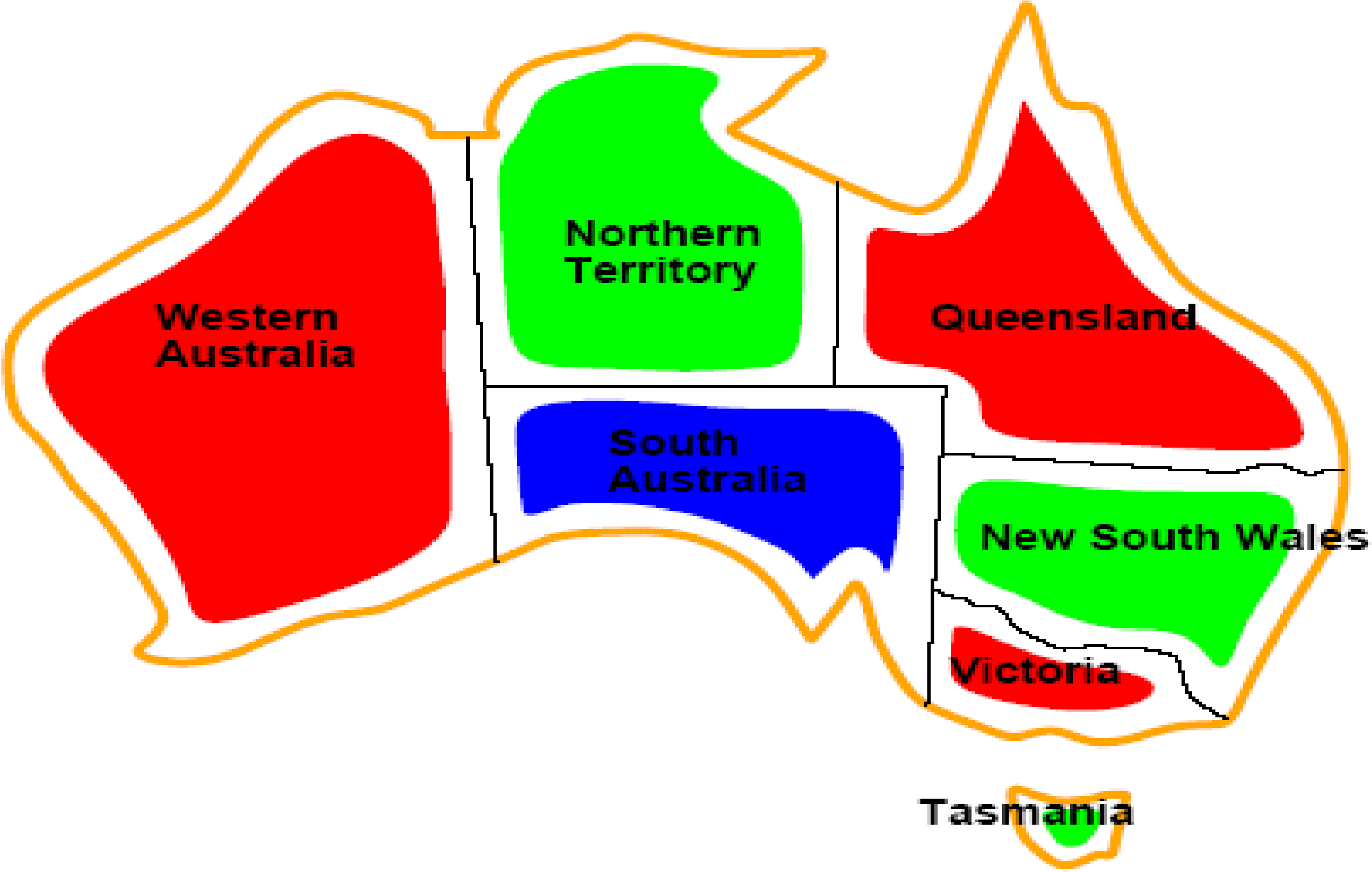
- Search for solutions to variables that **satisfy** constraints
- Rule out known-false candidates early
- Begin search intelligently
- Goal: find a **complete** and **consistent** solution

Example: Map Coloring



- Variables: *WA, NT, Q, NSW, V, SA, T*
- Domains: D_i=*{red,green,blue}*
- Constraints: adjacent regions must have different colors.
  - E.g. $WA \neq NT$

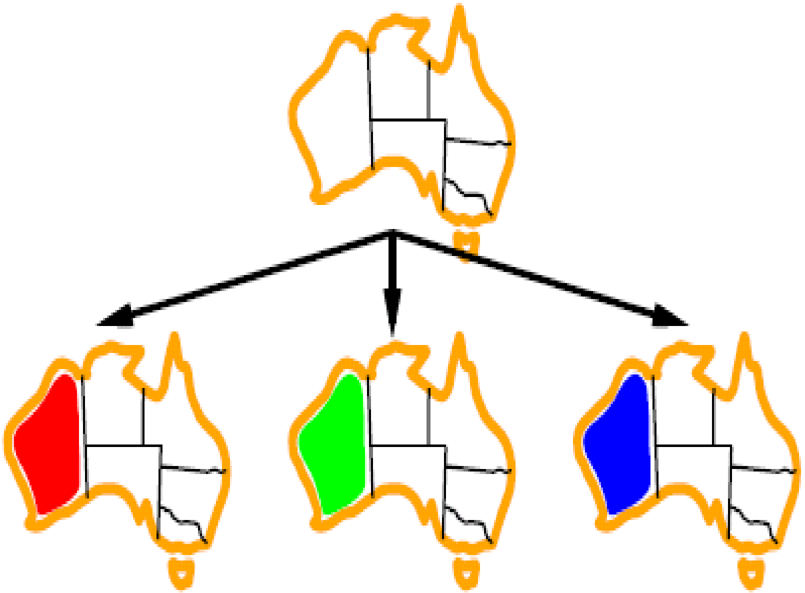
Example: Map Coloring



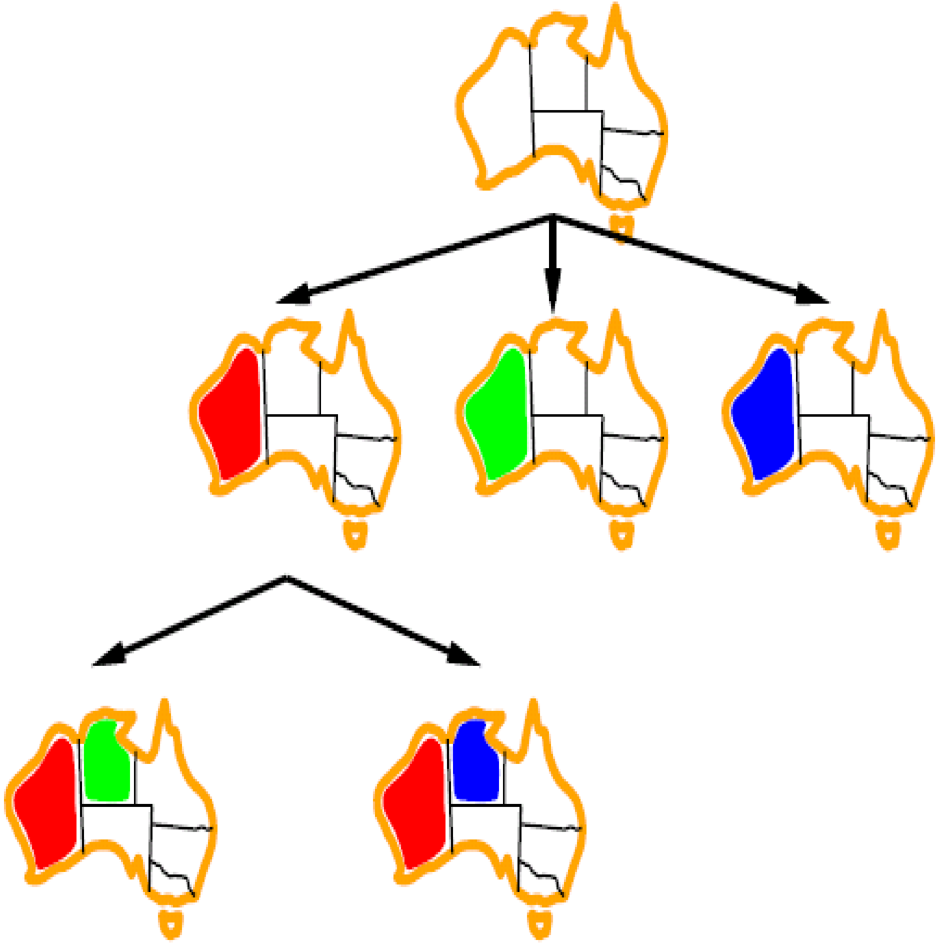
Backtracking Search

- Depth-first search
- Assign value to variable at each step
- Backtrack if conflict found

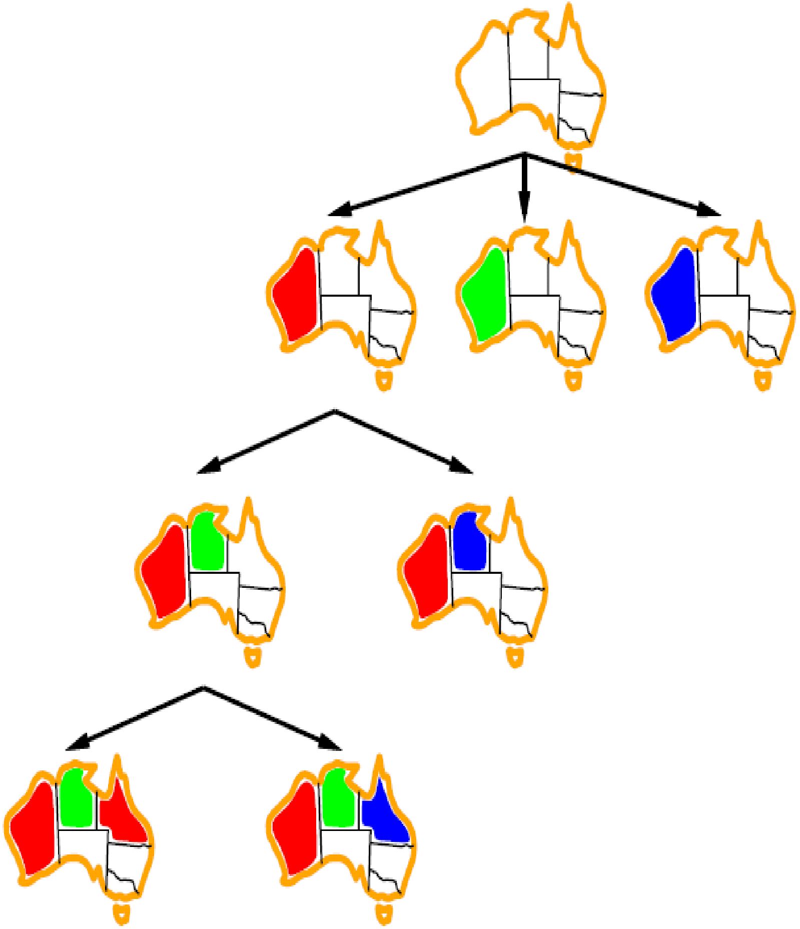
Backtracking Search



Backtracking Search



Backtracking Search

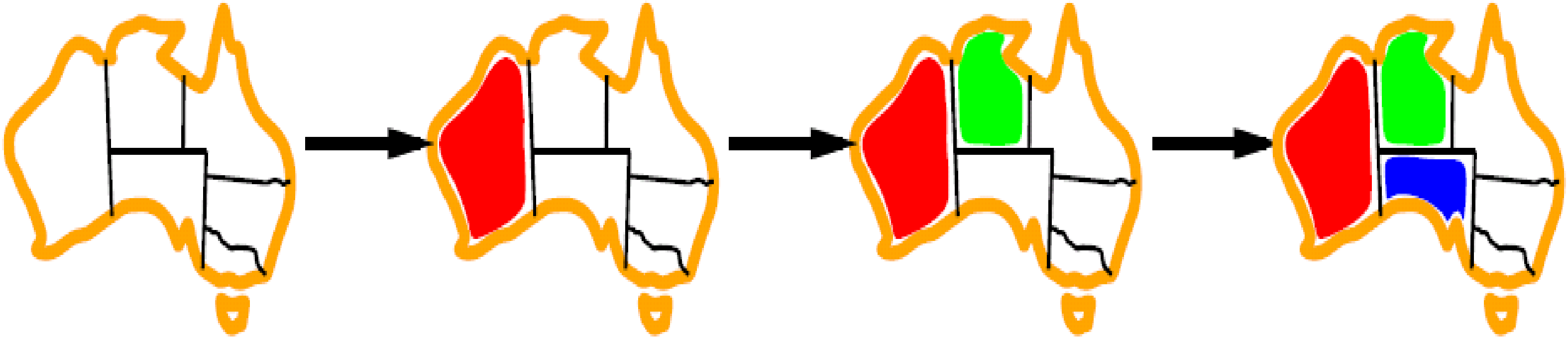


Improving Efficiency

- Depth-first search
- **Assign value to variable** at each step
- Backtrack if conflict found

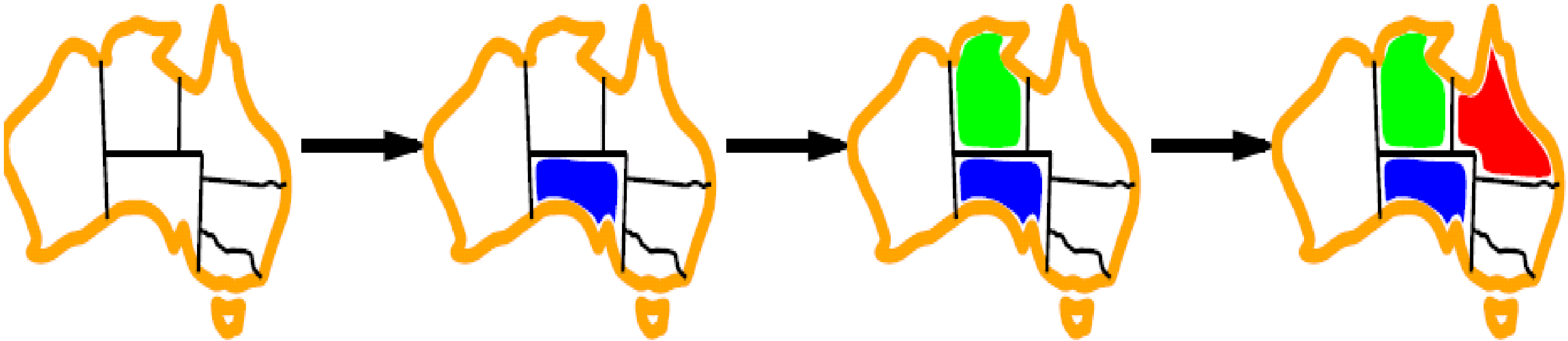
Variable Selection

Minimum Remaining Values



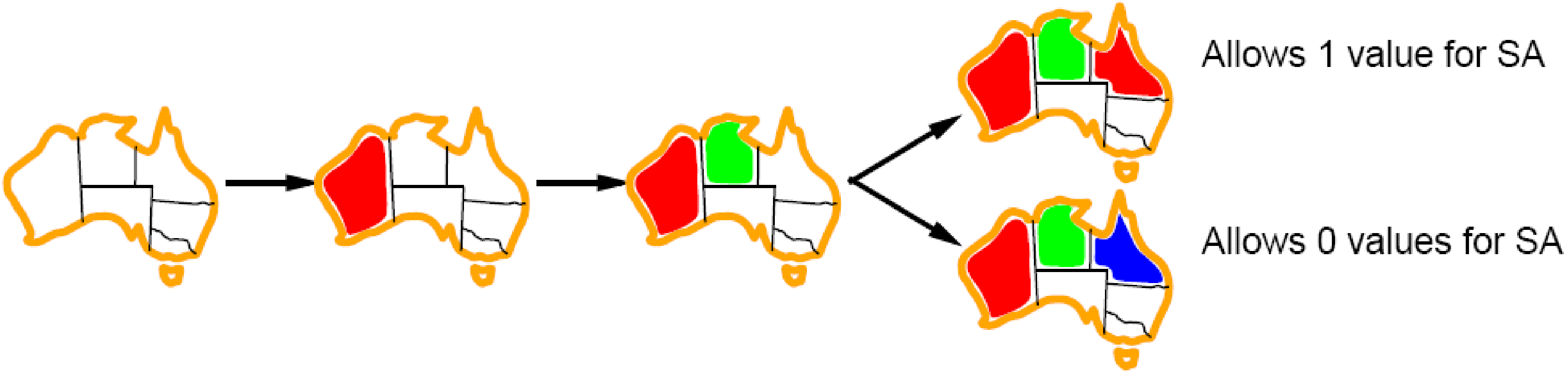
Variable Selection

Highest degree



Value Selection

Least constraining value



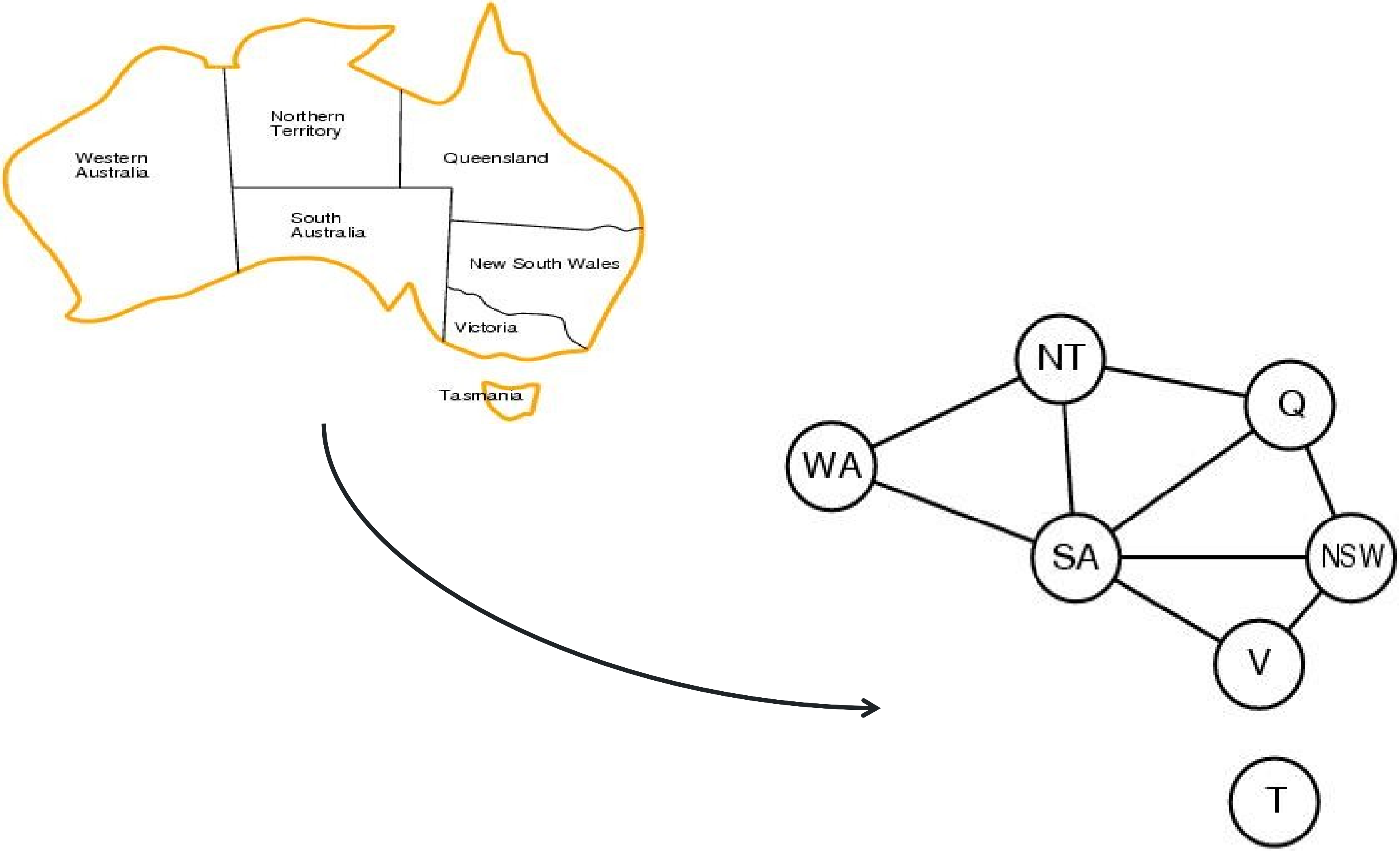
So what’s the improvement?

- Plain backtracking: 25-queens problem
- Backtracking with heuristics: 1000-queens

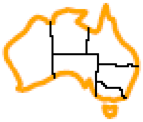
Improving Efficiency

- Depth-first search
- Assign value to variable at each step
- Backtrack **if conflict found**

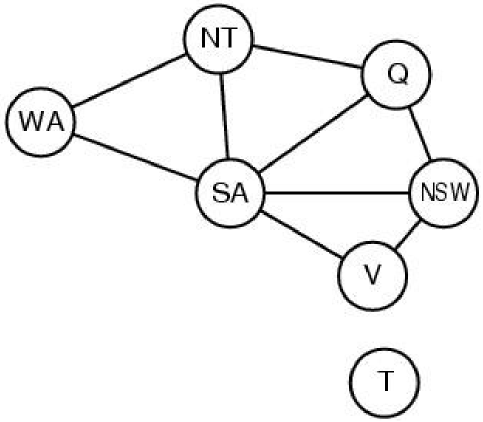
Binary Constraint Graphs



Forward Checking

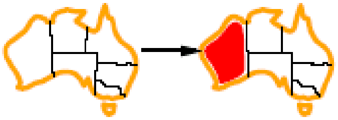


WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦

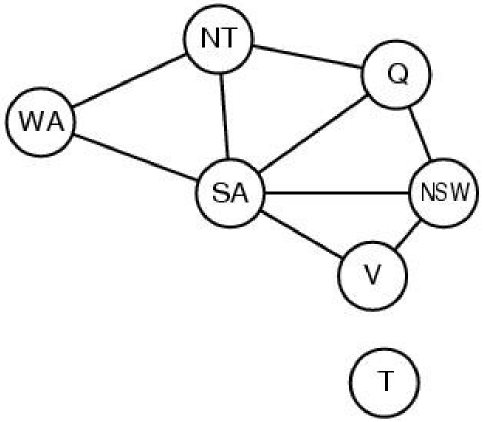


- Maintain list of remaining legal values for unassigned variables
- Conflict arises if there are no more values for any variable

Forward Checking



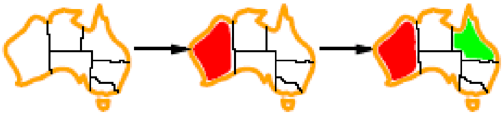
WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦
🟥	🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟩🟦	🟥🟩🟦



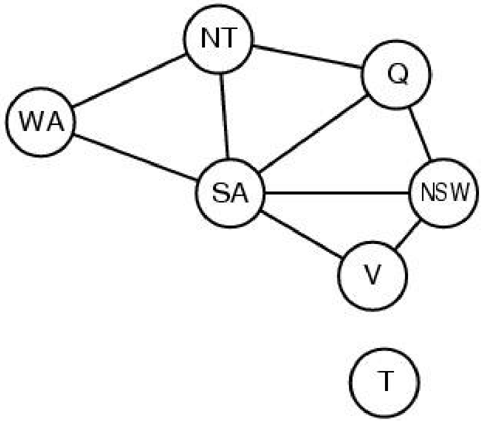
- Maintain list of remaining legal values for unassigned variables

- Conflict arises if there are no more values for any variable

Forward Checking



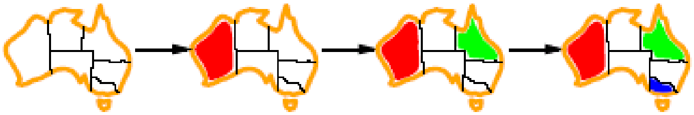
WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦
🟥	🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟩🟦	🟥🟩🟦
🟥	🟦	🟩	🟥🟦	🟥🟩🟦	🟦	🟥🟩🟦



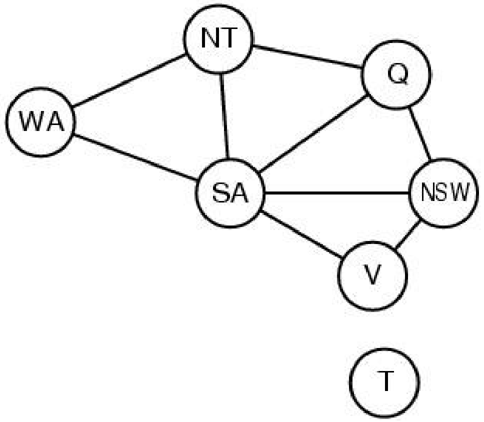
- Maintain list of remaining legal values for unassigned variables

- Conflict arises if there are no more values for any variable

Forward Checking

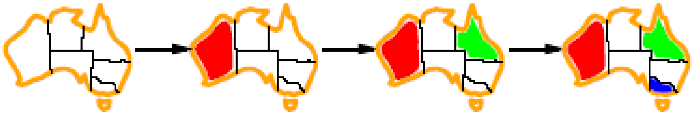


WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦
🟥	🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟩🟦	🟥🟩🟦
🟥	🟦	🟩	🟥🟦	🟥🟩🟦	🟦	🟥🟩🟦
🟥	🟦	🟩	🟥	🟦		🟥🟩🟦

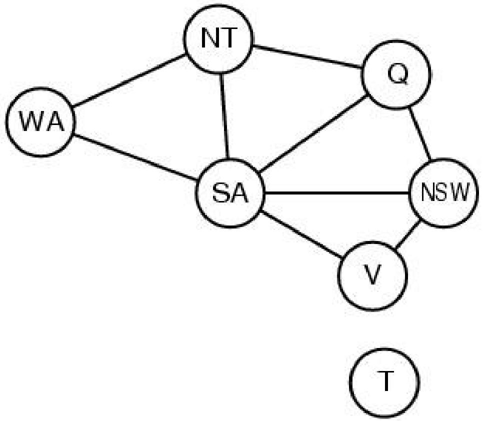


- Maintain list of remaining legal values for unassigned variables
- Conflict arises if there are no more values for any variable

Forward Checking



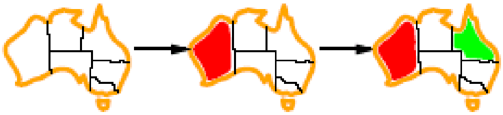
WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦
🟥	🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟩🟦	🟥🟩🟦
🟥	🟦	🟩	🟥🟦	🟥🟩🟦	🟦	🟥🟩🟦
🟥	🟦	🟩	🟥	🟦		🟥🟩🟦



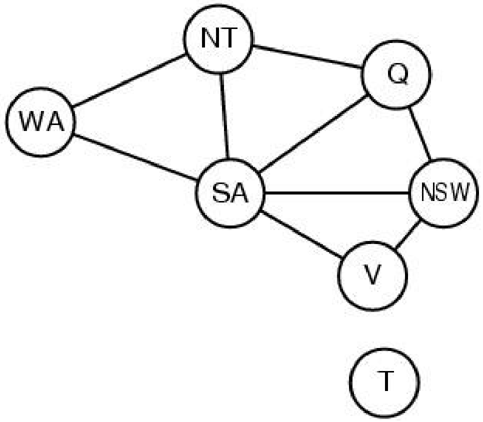
- Maintain list of remaining legal values for unassigned variables

- **Conflict arises** if there are no more values for any variable

But…



WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦
🟥	🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟩🟦	🟥🟩🟦
🟥	🟦	🟩	🟥🟦	🟥🟩🟦	🟦	🟥🟩🟦

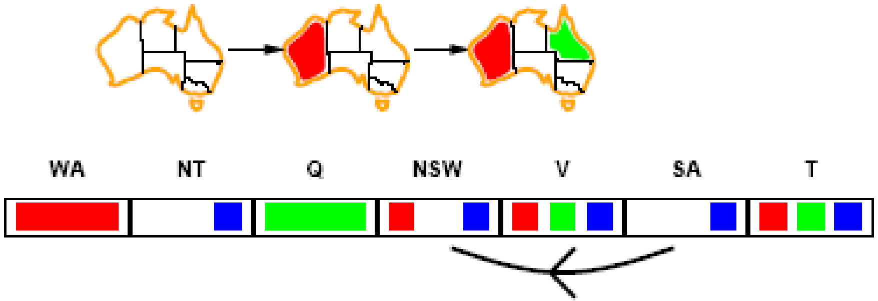


- Could we have known to stop the search earlier?

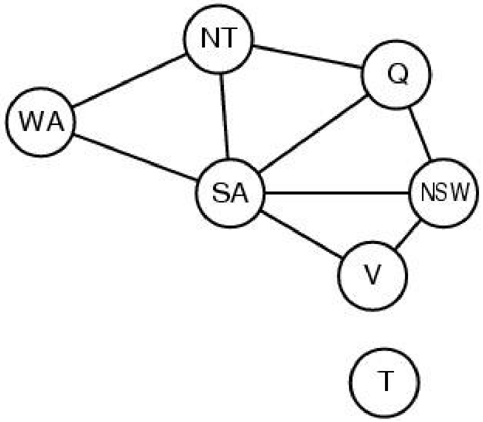
Arc Consistency

An arc (X, Y) in the graph is **consistent** if for every value of *a* of X there exists a value *b* of Y that is consistent with a.

Arc Consistency

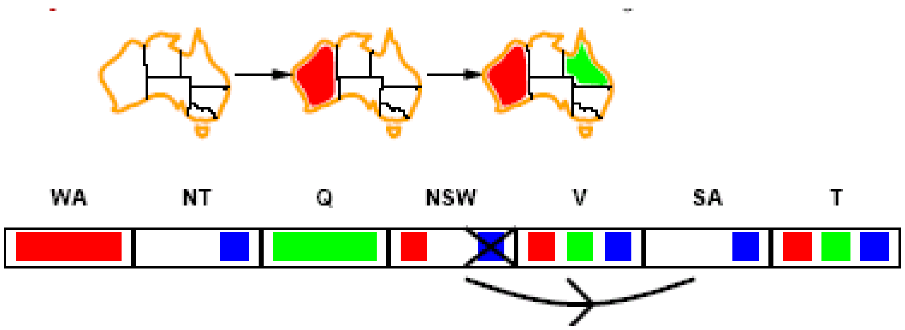


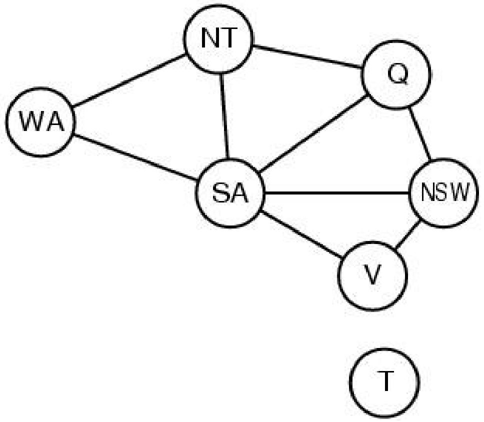
WA	NT	Q	NSW	V	SA	T
🟥	🟦	🟩	🟥 🟦	🟥 🟩 🟦	🟦	🟥 🟩 🟦



- (SA, NSW) is consistent if:
  - SA = blue, NSW = red

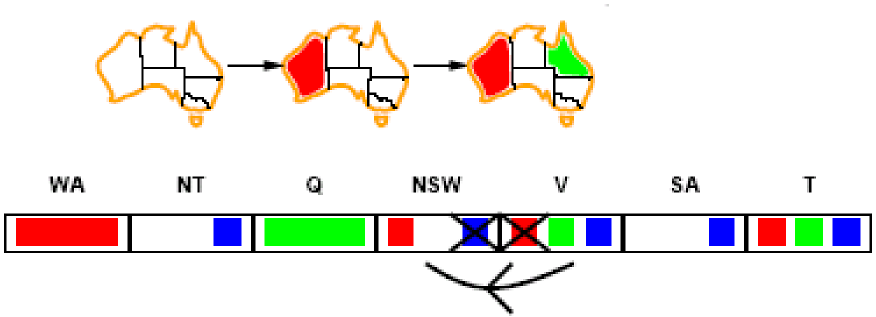
Arc Consistency

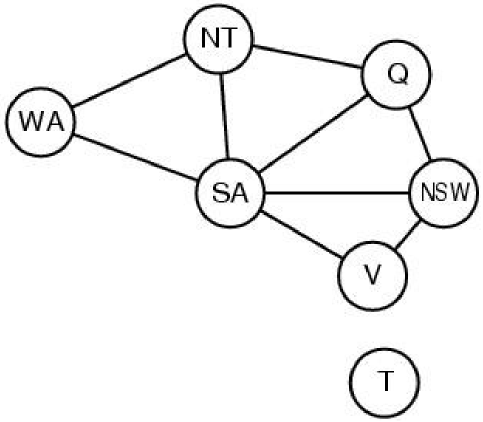




- (NSW, SA) is consistent if:
    - NSW = red, SA = blue
    - NSW = blue, SA = ☹

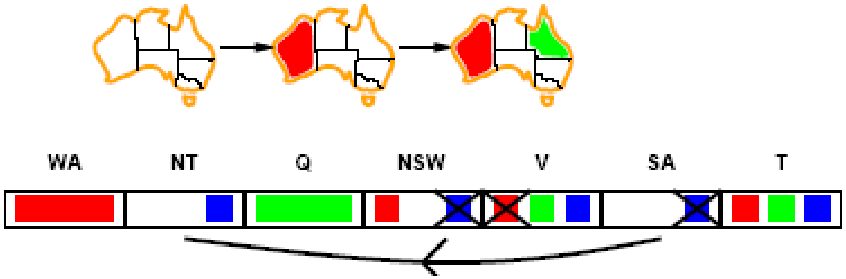
Arc Consistency

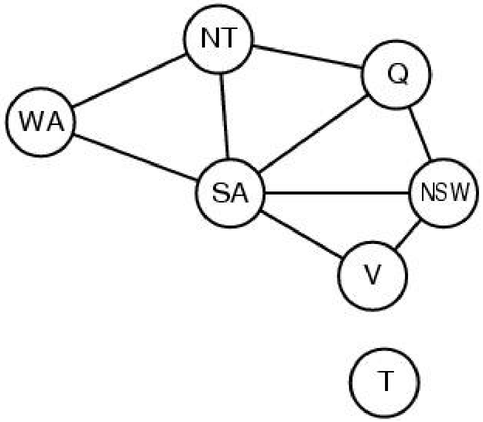




- (V, NSW) is consistent if:
    - V is not red

Arc Consistency





- (SA, NT) is not consistent

Arc Consistency: Tradeoffs

- Lots of overhead
  - Need to re-add an edge to the queue if you change its values elsewhere in the check
  - Checking every edge on every step of search
- Vastly reduces search space

APPLICATION AREA:
PROCEDURAL CONTENT GENERATION

What is Procedural Content Generation?

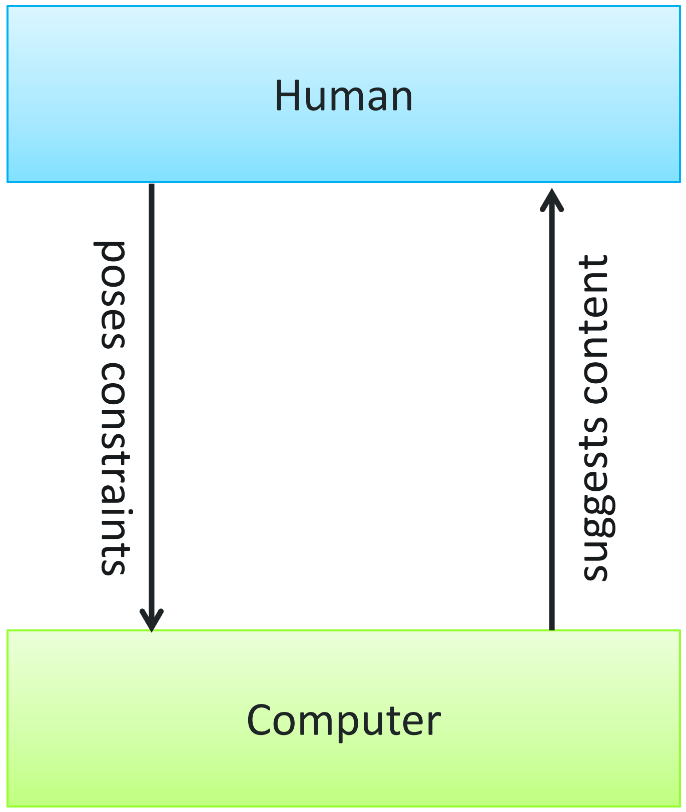
The **programmatic creation** of content
that has a **meaningful impact** on gameplay
using algorithms that **understand** games and players.

Kinds of PCG Use

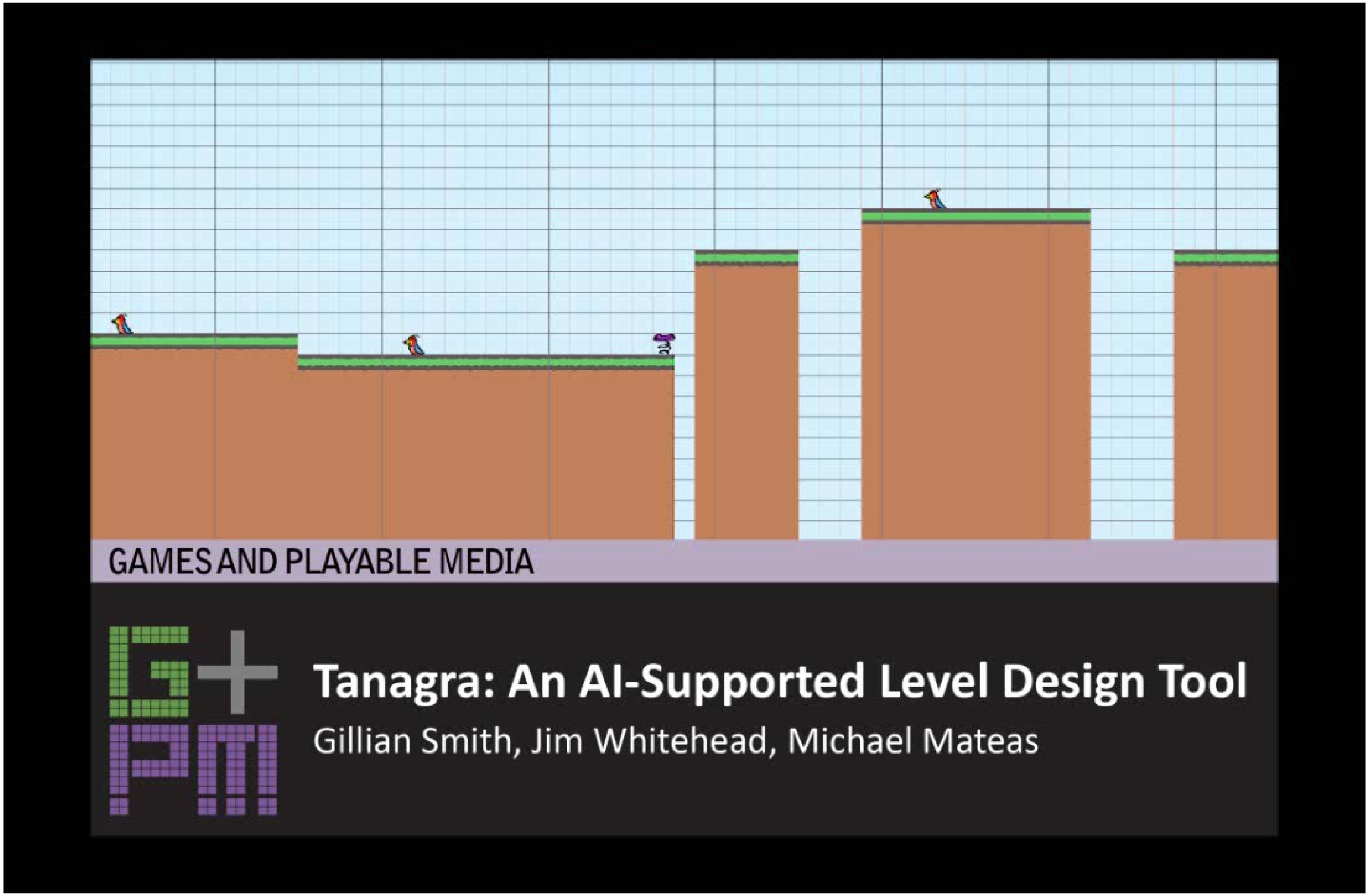
- Data compression
- Replayability
- Enabling exploration

A Mixed-Initiative Approach

Taking turns designing a level with the computer



Tanagra

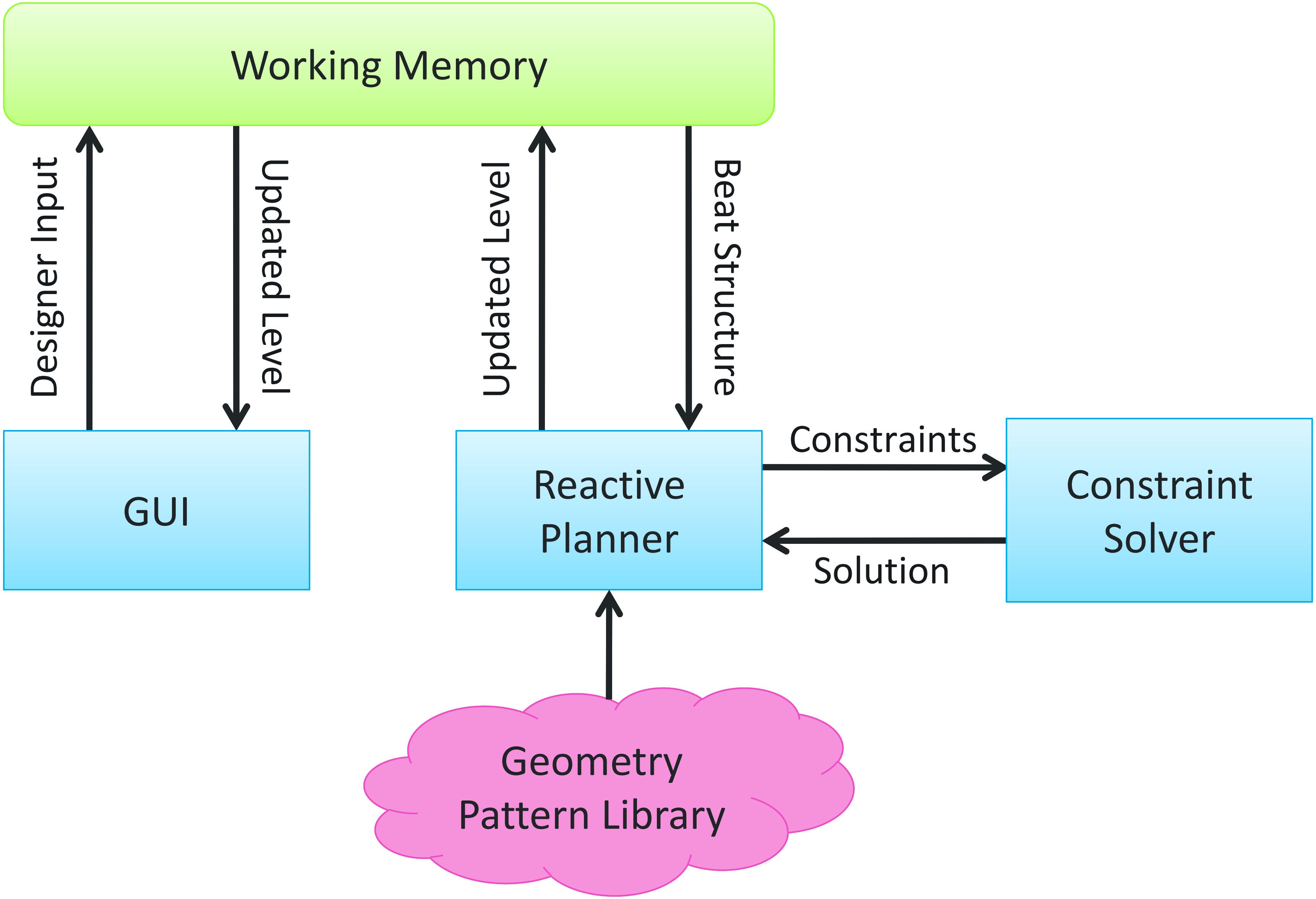


GAMES AND PLAYABLE MEDIA

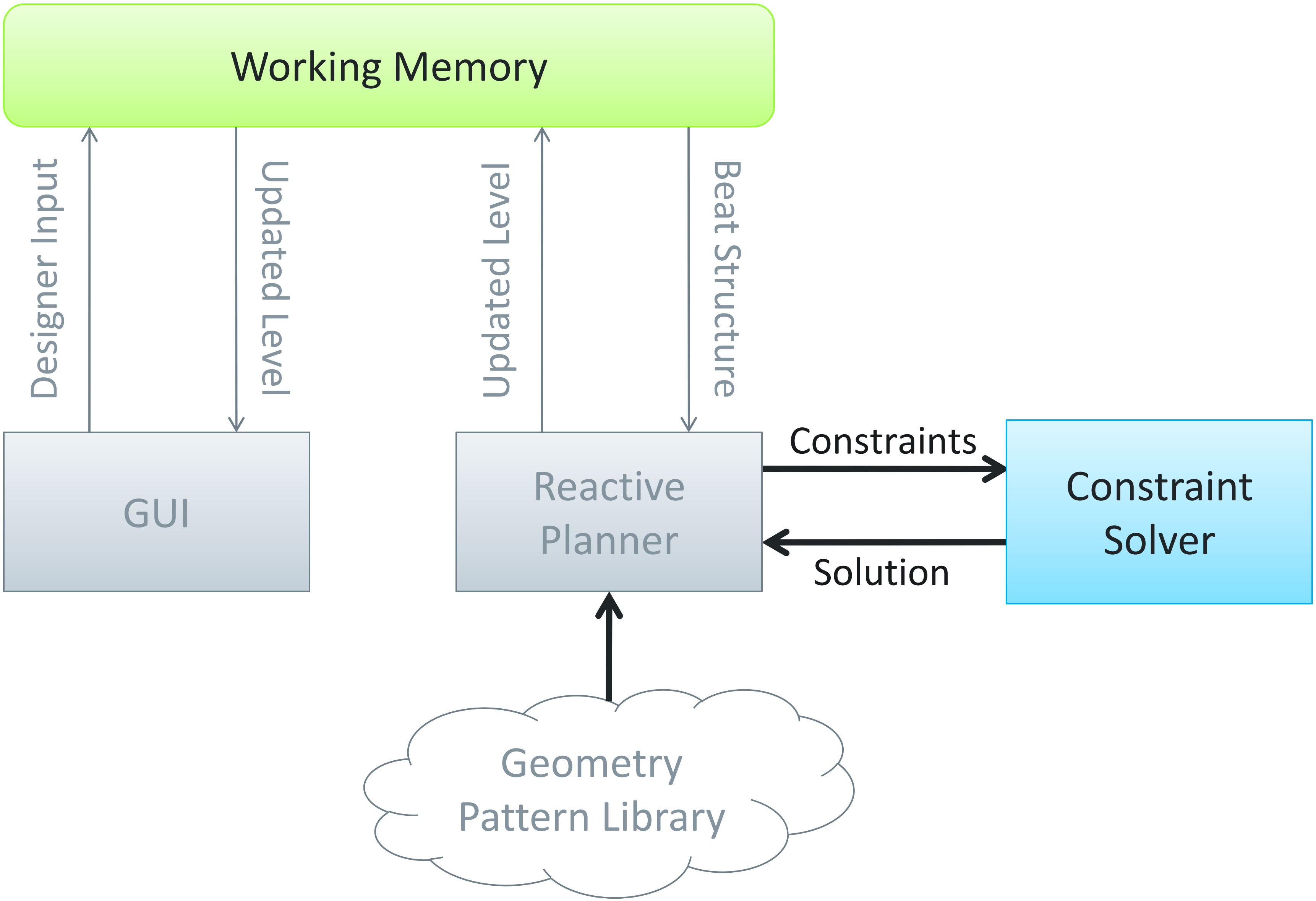
Tanagra: An AI-Supported Level Design Tool

Gillian Smith, Jim Whitehead, Michael Mateas

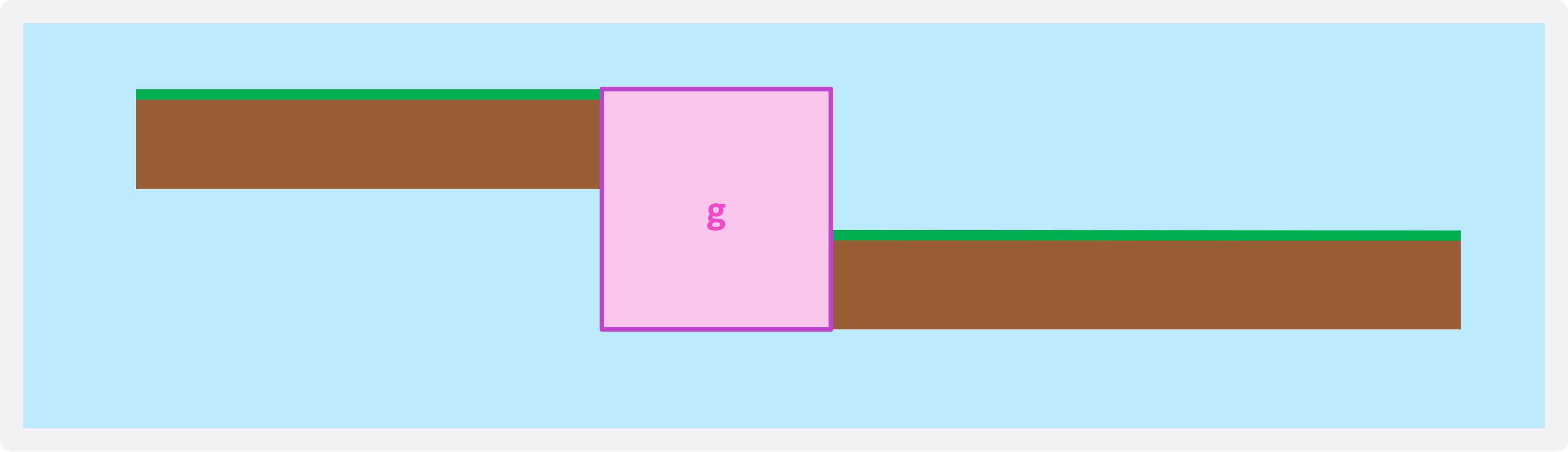
Tanagra Architecture



Constraint Solving with Choco



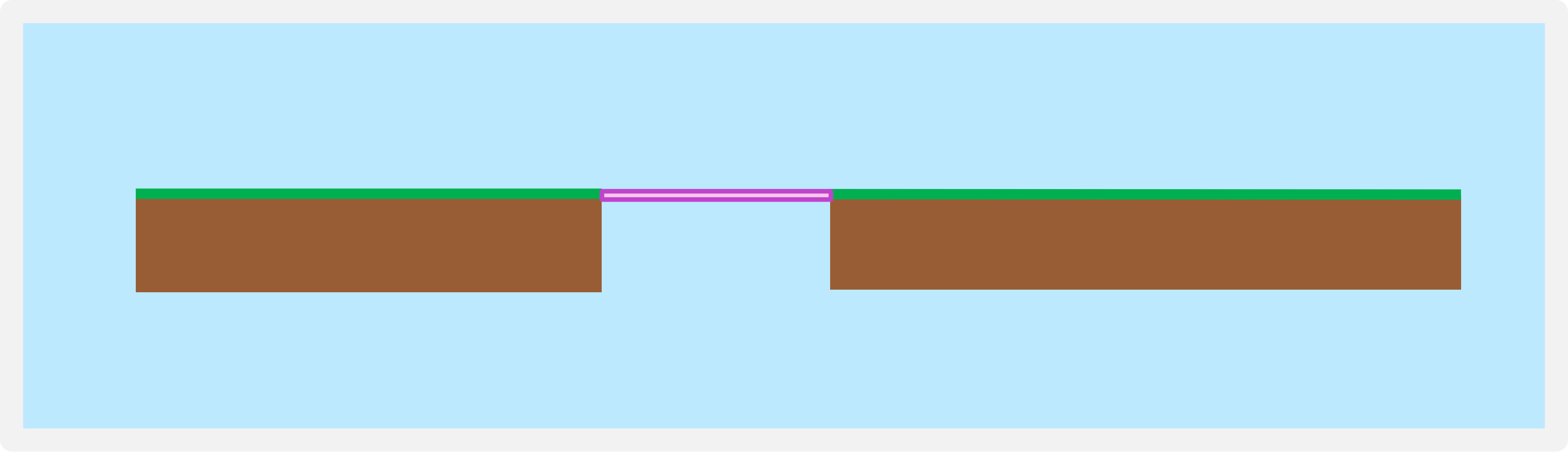
Constraints



g.height() == 0 → g.width() != 0

g.width() == 0 → g.height() != 0

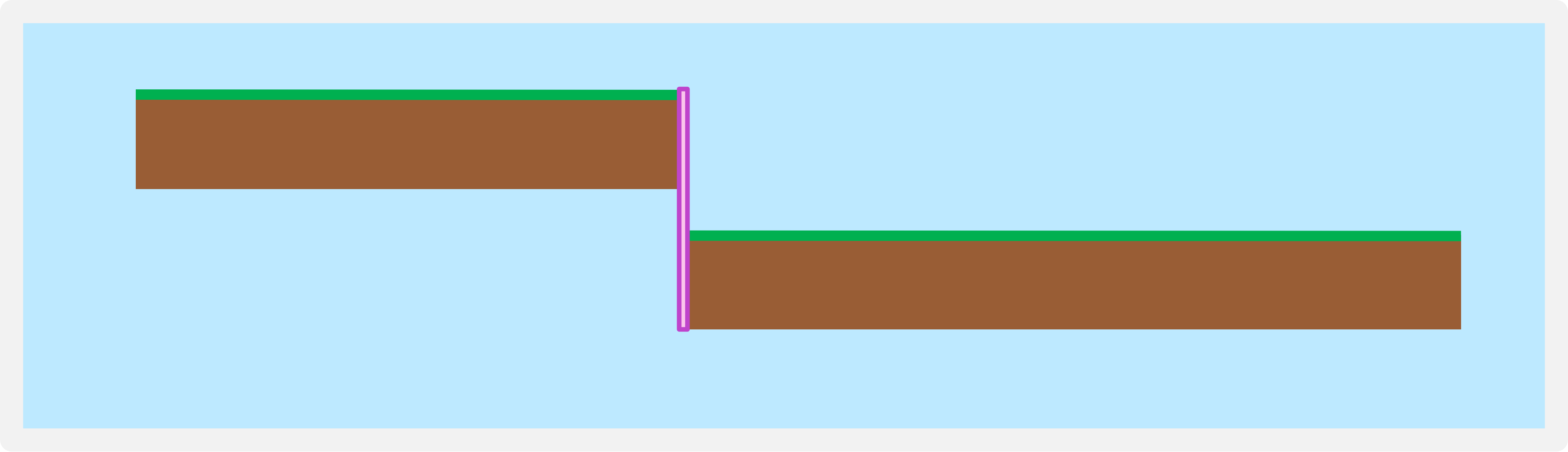
Constraints



g.height() == 0 → g.width() != 0

g.width() == 0 → g.height() != 0

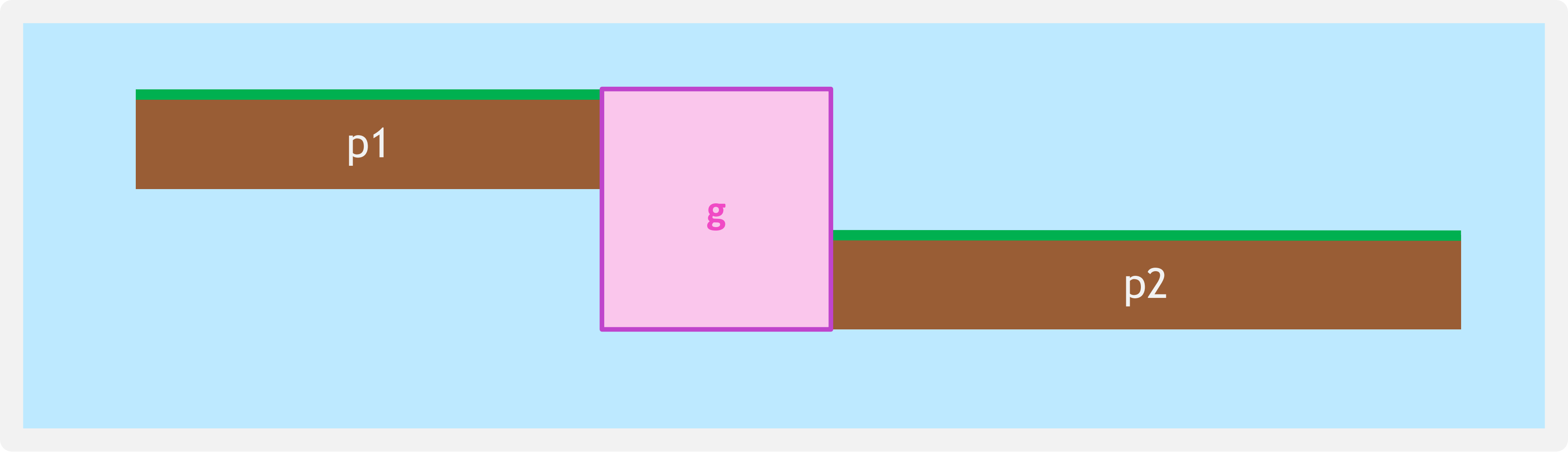
Constraints



g.height() == 0 → g.width() != 0

g.width() == 0 → g.height() != 0

Constraints

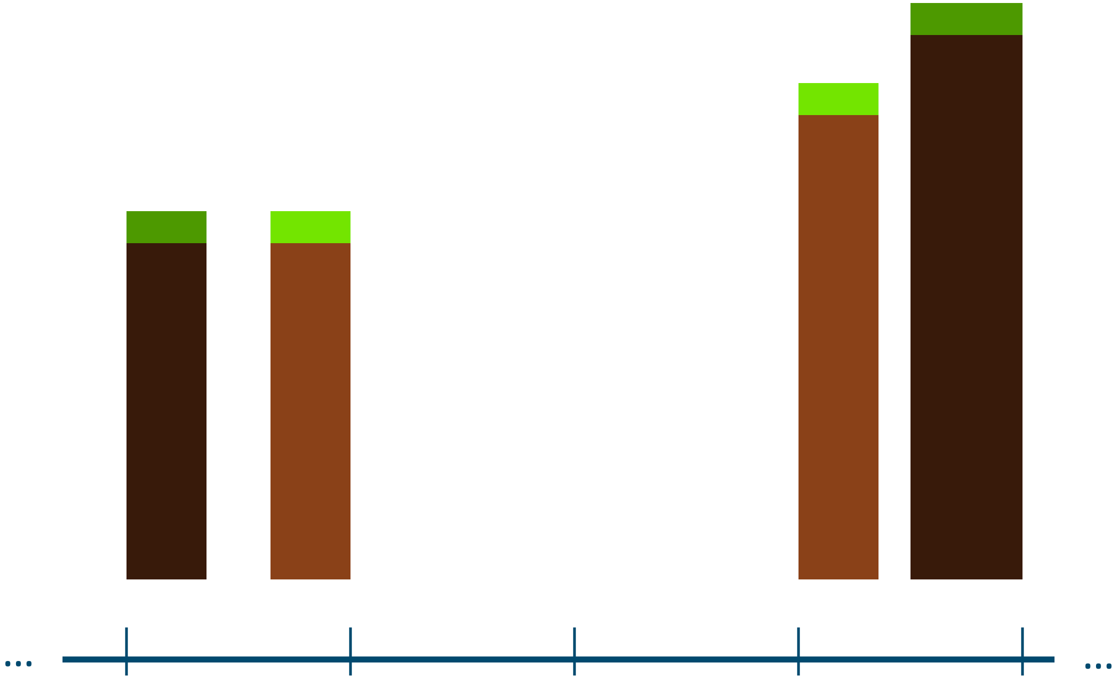


p2.startX() = p1.endX() + g.width()

p2.startY() = p1.endY() + g.height()

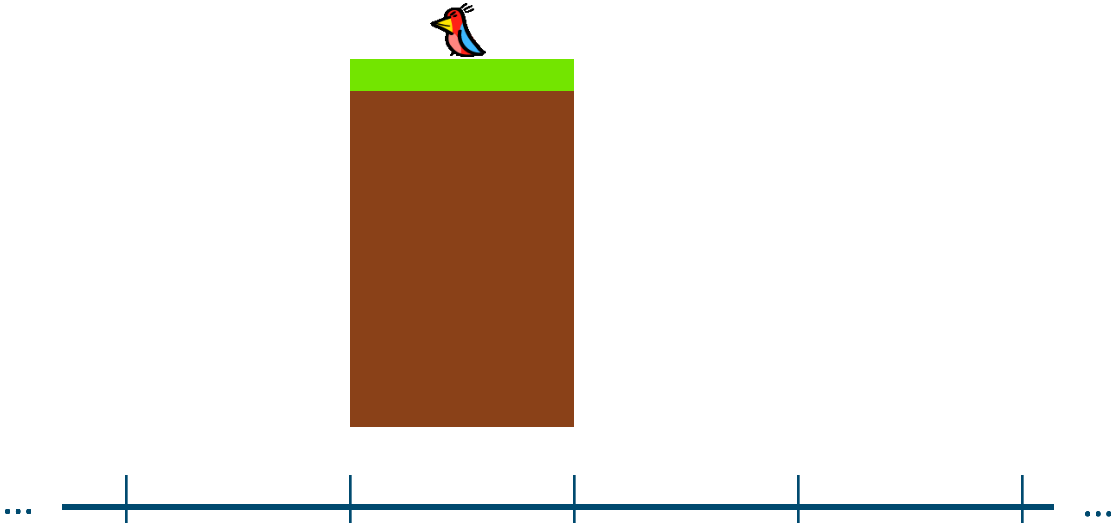
Finding a Solution

- Solve constraints after placing all geometry
  - Choose geometry intelligently based on surroundings
- If no solution:
  - Remove positioning constraints and retry
  - If no solution:
    - Mark geometry combination as invalid and attempt a different pattern



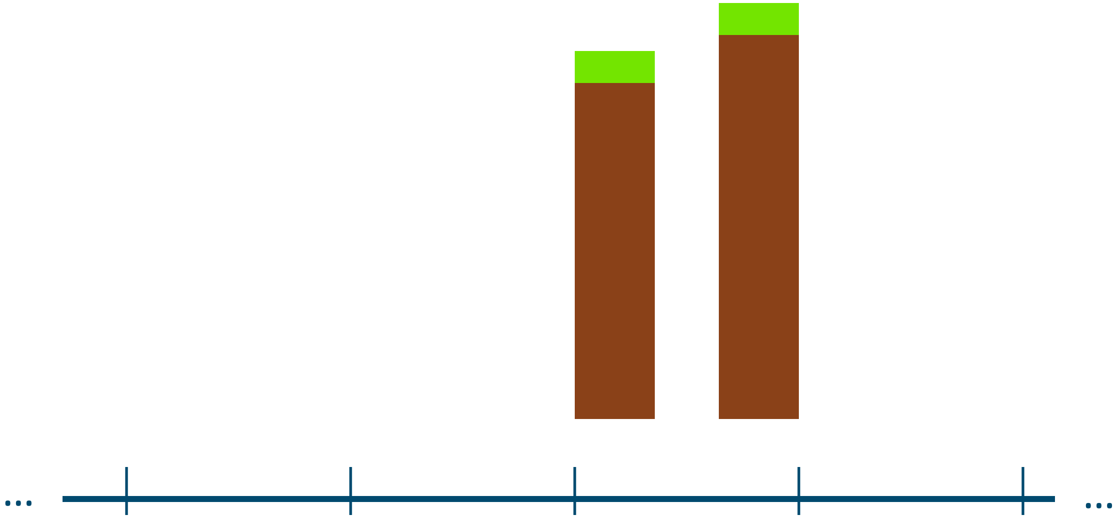
Finding a Solution

- Solve constraints after placing all geometry
  - Choose geometry intelligently based on surroundings
- If no solution:
  - Remove positioning constraints and retry
  - If no solution:
    - Mark geometry combination as invalid and attempt a different pattern



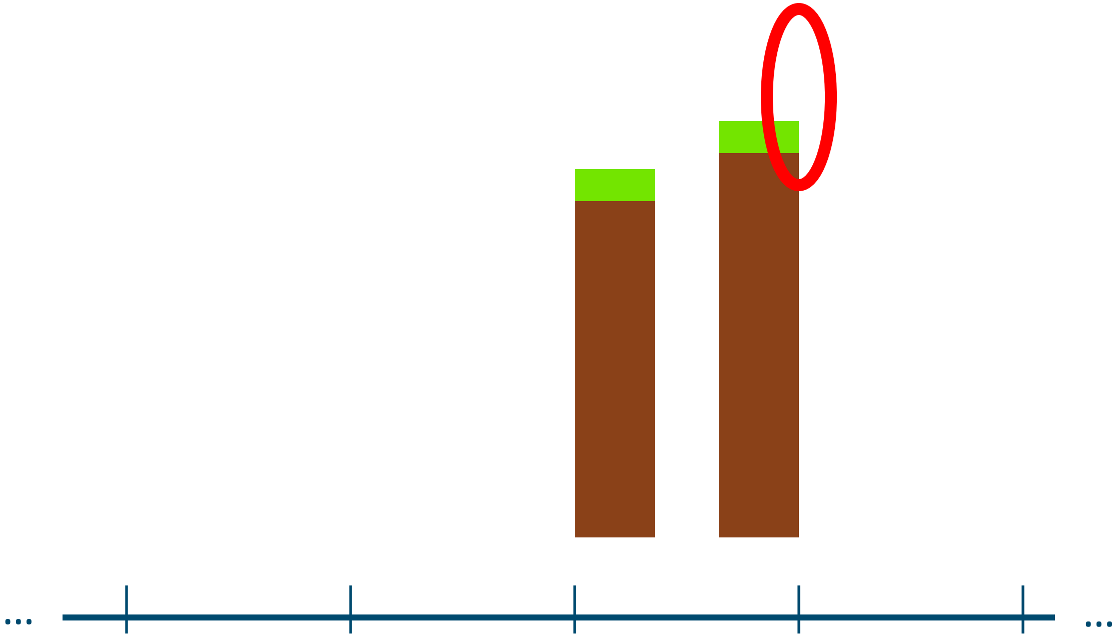
Finding a Solution

- Solve constraints after placing all geometry
  - Choose geometry intelligently based on surroundings
- If no solution:
  - Remove positioning constraints and retry
  - If no solution:
    - Mark geometry combination as invalid and attempt a different pattern



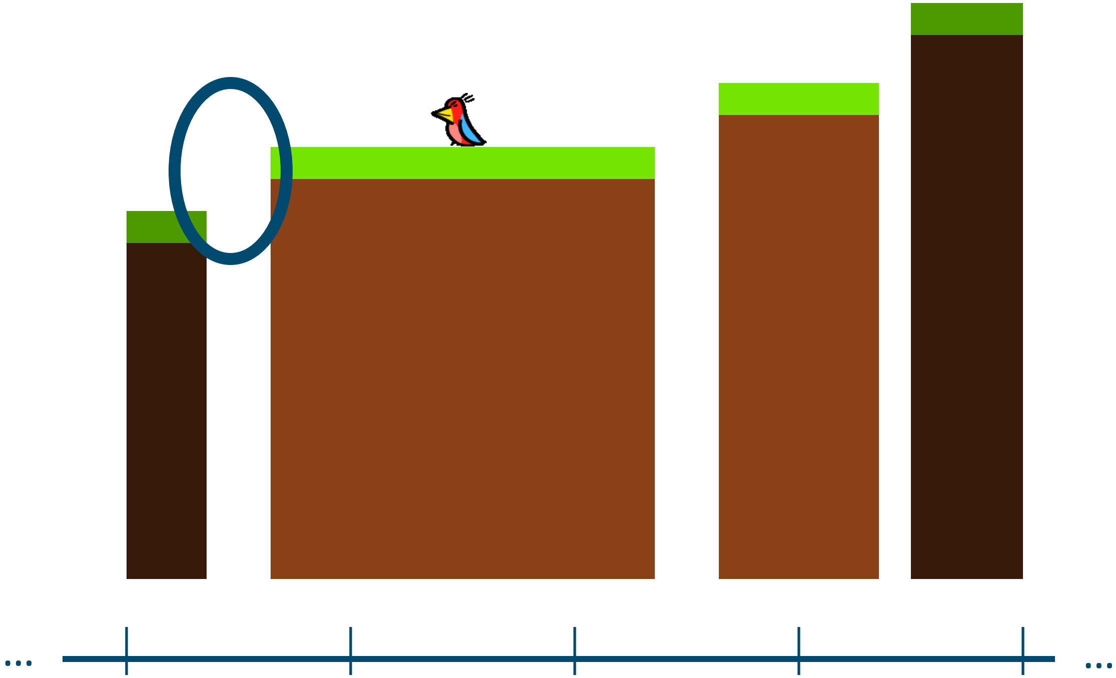
Finding a Solution

- Solve constraints after placing all geometry
  - Choose geometry intelligently based on surroundings
- If no solution:
  - Remove positioning constraints and retry
  - If no solution:
    - Mark geometry combination as invalid and attempt a different pattern



Finding a Solution

- Solve constraints after placing all geometry
  - Choose geometry intelligently based on surroundings
- If no solution:
  - **Remove positioning constraints and retry**
  - If no solution:
    - Mark geometry combination as invalid and attempt a different pattern



Constraint Programming for PCG

- Declarative representation
- Algorithm-agnostic
- Adding and removing constraints to shape generative space

ANSWER SET PROGRAMMING