Constraint Satisfaction

CSCI 4511/6511

Joe Goldfrank

Announcements

- Homework 2 is due on 21 February at 11:55 PM
- Homework 3 is released
    - Working with one partner is optionally permitted

Games

Minimax

- Initial state s_0
- $\textsc{Actions}(s)$ and $\textsc{To-move}(s)$
- $\textsc{Result}(s, a)$
- $\textsc{Is-Terminal}(s)$
- $\textsc{Utility}(s, p)$

Minimax

Games of Luck

- Real-world problems are rarely deterministic
- Non-deterministic state evolution:
    - Roll a die to determine next position
    - Toss a coin to determine who picks candy first
    - Precise trajectory of kicked football[1]
    - Others?

---

[1] Any definition of "football"

Solving Non-Deterministic Games

Previously: Max and Min alternate turns

Now:

- Max
- Chance
- Min
- Chance

😣

We Played Another Game

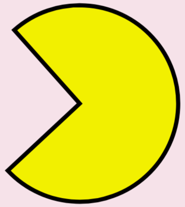
- Place 11 pieces of candy between you
- Alternating turns:
    - Choose to take one or two pieces
- *Except:*
    - After choosing, flip two coins, record *total* number of heads[1]
    - If total is divisible by 3, take one less piece than you chose
    - If total is divisible by 5, take one more piece than you chose
    - If total divisible by 15, take no candy
- Last person to take a piece wins all of the candy

---

[1] Keep a running total through the game

Expectiminimax

- “Expected value” of next position



- How does this impact branching factor of the search?

🫠

Filled With Uncertainty

What is to be done?

- Pruning is still possible
    - How?
- Heuristic evaluation functions
    - Choose carefully!

Non-Optimal Adversaries

- Is deterministic “best” behavior optimal?
- Are all adversaries rational?

- Expectimax

CSPs

Factored Representation

- Encode relationships between variables and states
- Solve problems with *general* search algorithms
  - Heuristics do not require expert knowledge of problem
  - Encoding problem requires expert knowledge of problem[1]

Why?

---

[1] But it always does.

Constraint Satisfaction

- Express problem in terms of state variables
  - Constrain state variables
- Begin with all variables unassigned
- Progressively assign values to variables
- Assignment of values to state variables that “works:” *solution*

More Formally

- State variables: $X_1, X_2, \ldots, X_n$
- State variable domains: $D_1, D_2, \ldots, D_n$
  - The domain specifies which values are permitted for the state variable
  - Domain: set of allowable variables (or permissible range for continuous variables)[1]
  - Some constraints $C_1, C_2, \ldots, C_m$ restrict allowable values

---

[1] Or a hybrid, such as a union of ranges of continuous variables.

Constraint Types

- Unary: restrict single variable
    - Can be rolled into domain
    - Why even have them?
- Binary: restricts two variables
- Global: restrict “all” variables

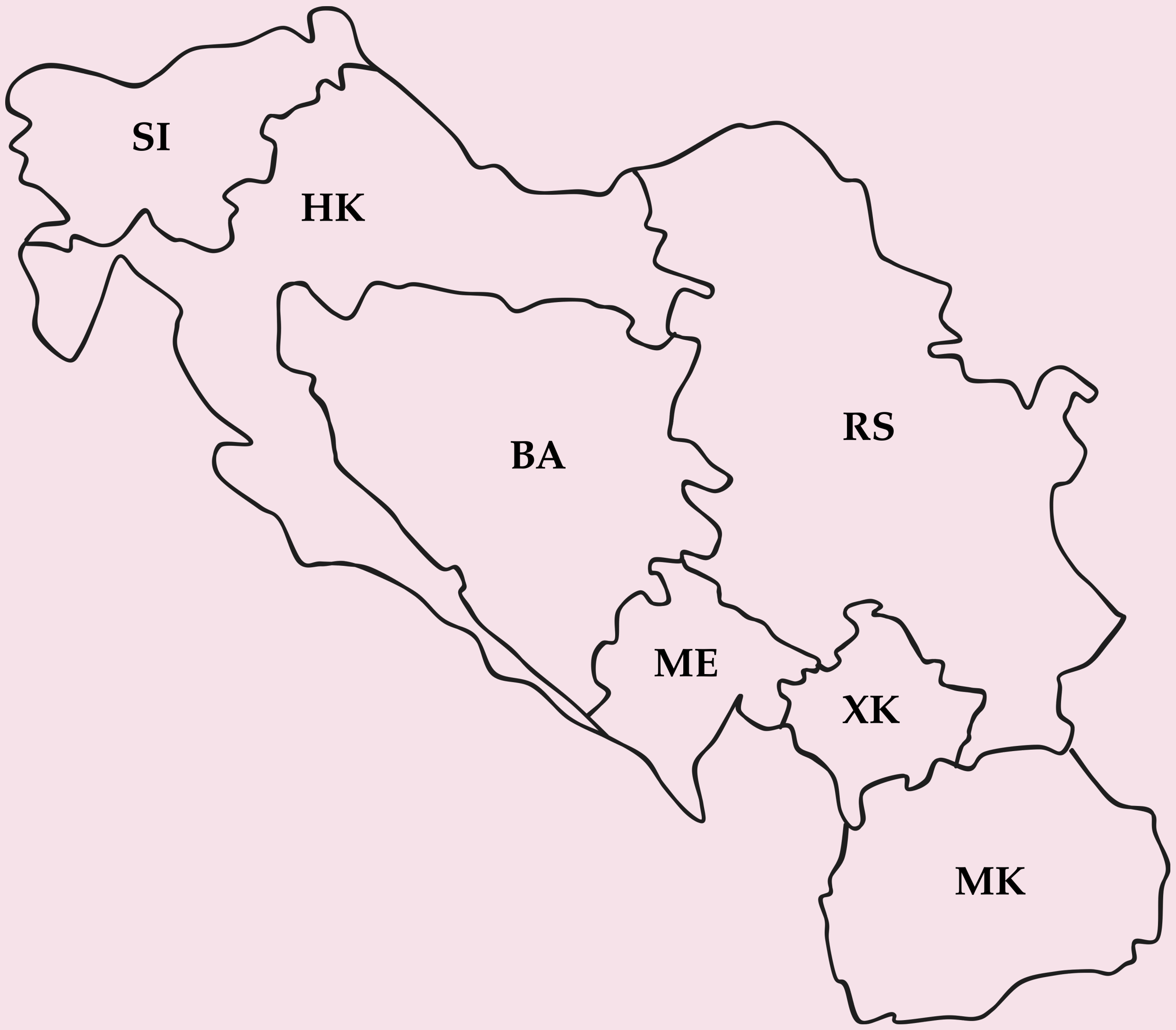
Constraint Examples

- X_1 and X_2 both have real domains, i.e. $X_1, X_2 \in \mathbb{R}$
  - A constraint could be $X_1 < X_2$
- X_1 could have domain $\{\text{red}, \text{green}, \text{blue}\}$ and X_2 could have domain $\{\text{green}, \text{blue}, \text{orange}\}$
  - A constraint could be $X_1 \neq X_2$
- $X_1, X_2, \ldots, X_100 \in \mathbb{R}$
  - Constraint: exactly four of X_i equal 12
  - Rewrite as binary constraint?

Assignments

- Assignments must be to values in each variable’s domain
- Assignment violates constraints?
    - Consistency
- All variables assigned?
    - Complete

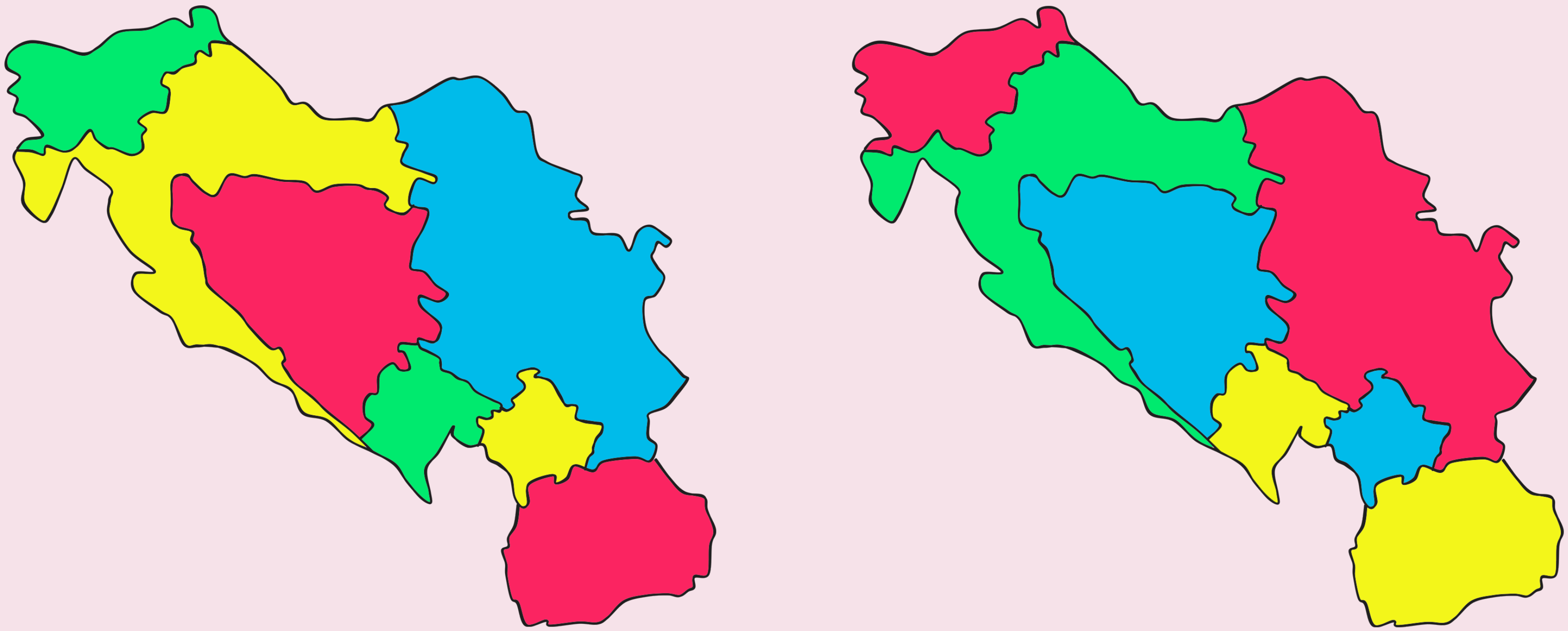
Yugoslavia[1]



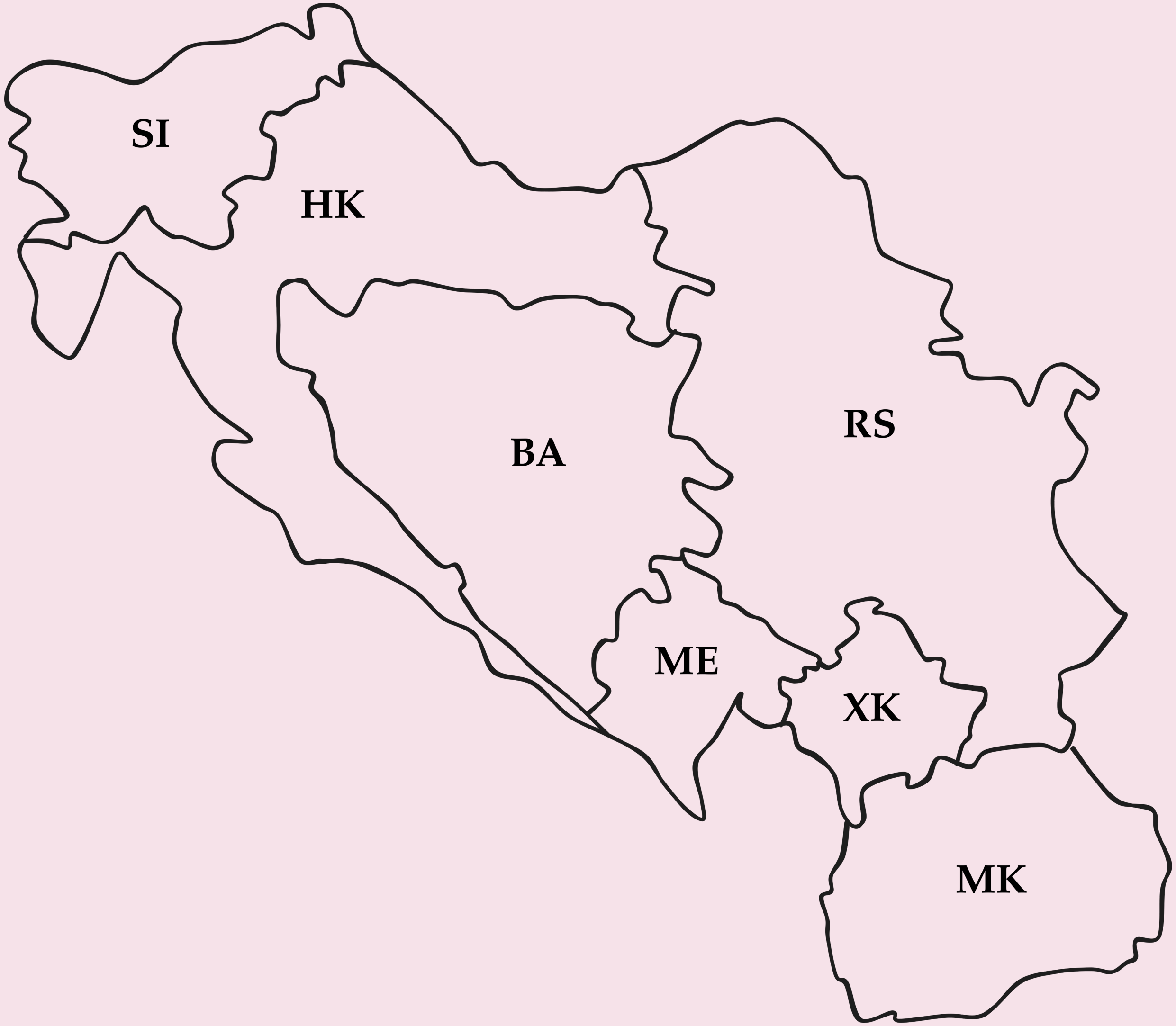
1. One of the most difficult problems of the 20th century

Four-Colorings

Two possibilities:



Formulate as CSP?



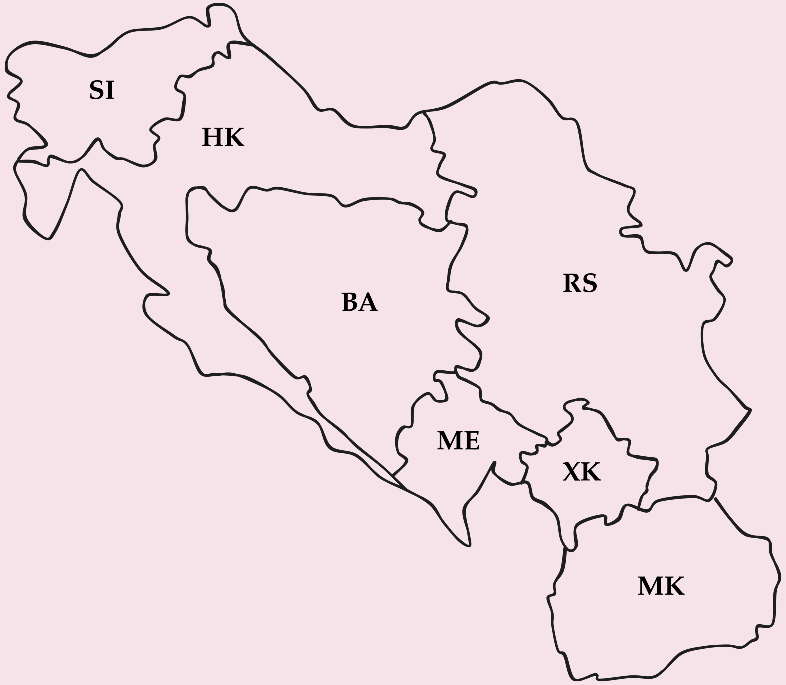
Graph Representations

- Constraint graph:
  - Nodes are variables
  - Edges are constraints
- Constraint hypergraph:
  - Variables are nodes
  - Constraints are nodes
  - Edges show relationship

Why have two different representations?

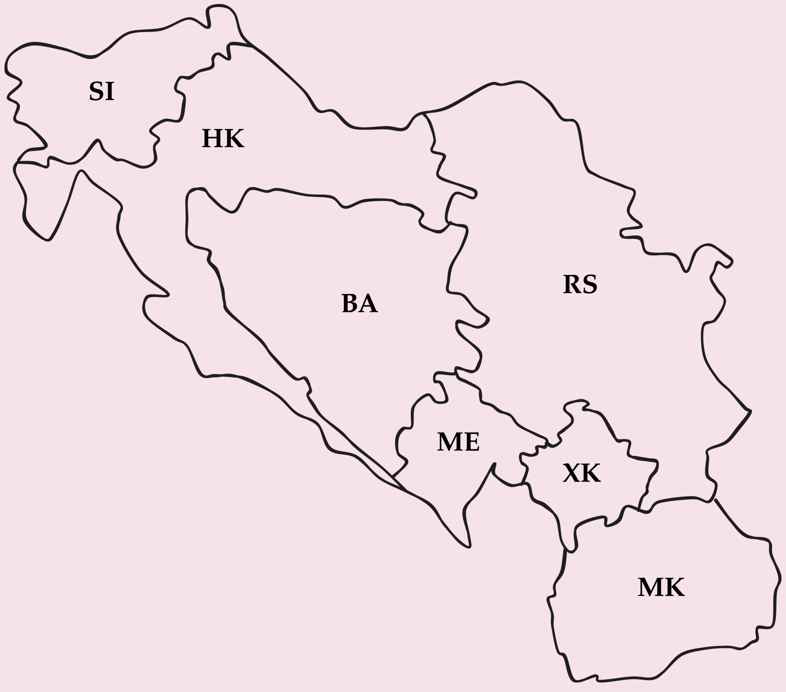
Graph Representation I

Constraint graph: edges are constraints



Graph Representation II

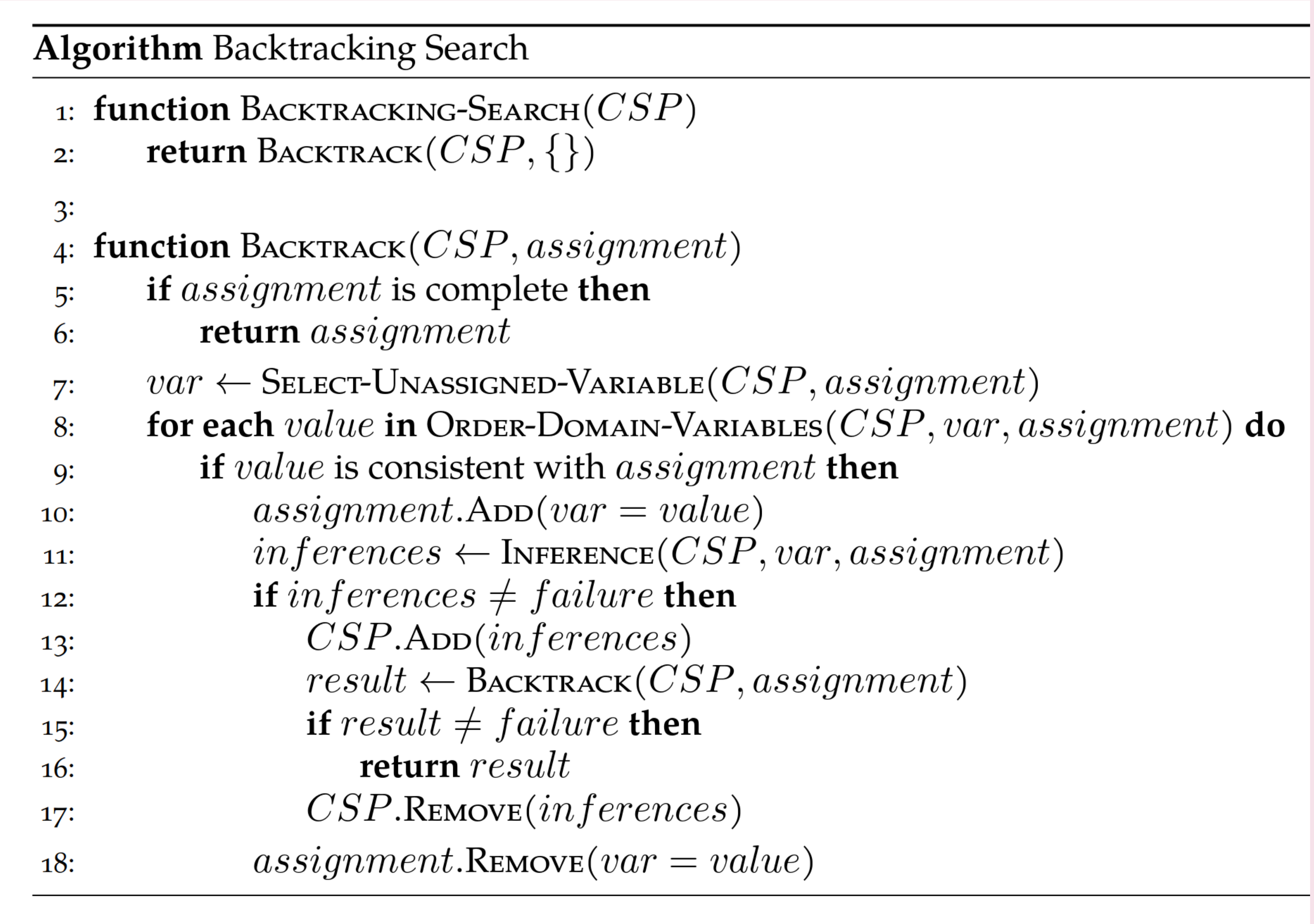
Constraint hypergraph: constraints are nodes



How To Solve It

- We can search!
  - …the space of consistent assignments
- Complexity $O(d^n)$
  - Domain size d, number of nodes n
- Tree search for node assignment
  - Inference to reduce domain size
- Recursive search

How To Solve It



Algorithm Backtracking Search

```
function BACKTRACKING-SEARCH($CSP$)
    return BACKTRACK($CSP, \{\}$)

function BACKTRACK($CSP, assignment$)
    if $assignment$ is complete then
        return $assignment$
    $var \leftarrow$ SELECT-UNASSIGNED-VARIABLE($CSP, assignment$)
    for each $value$ in ORDER-DOMAIN-VARIABLES($CSP, var, assignment$) do
        if $value$ is consistent with $assignment$ then
            $assignment$.ADD($var = value$)
            $inferences \leftarrow$ INFERENCE($CSP, var, assignment$)
            if $inferences \neq failure$ then
                $CSP$.ADD($inferences$)
                $result \leftarrow$ BACKTRACK($CSP, assignment$)
                if $result \neq failure$ then
                    return $result$
                $CSP$.REMOVE($inferences$)
            $assignment$.REMOVE($var = value$)
```

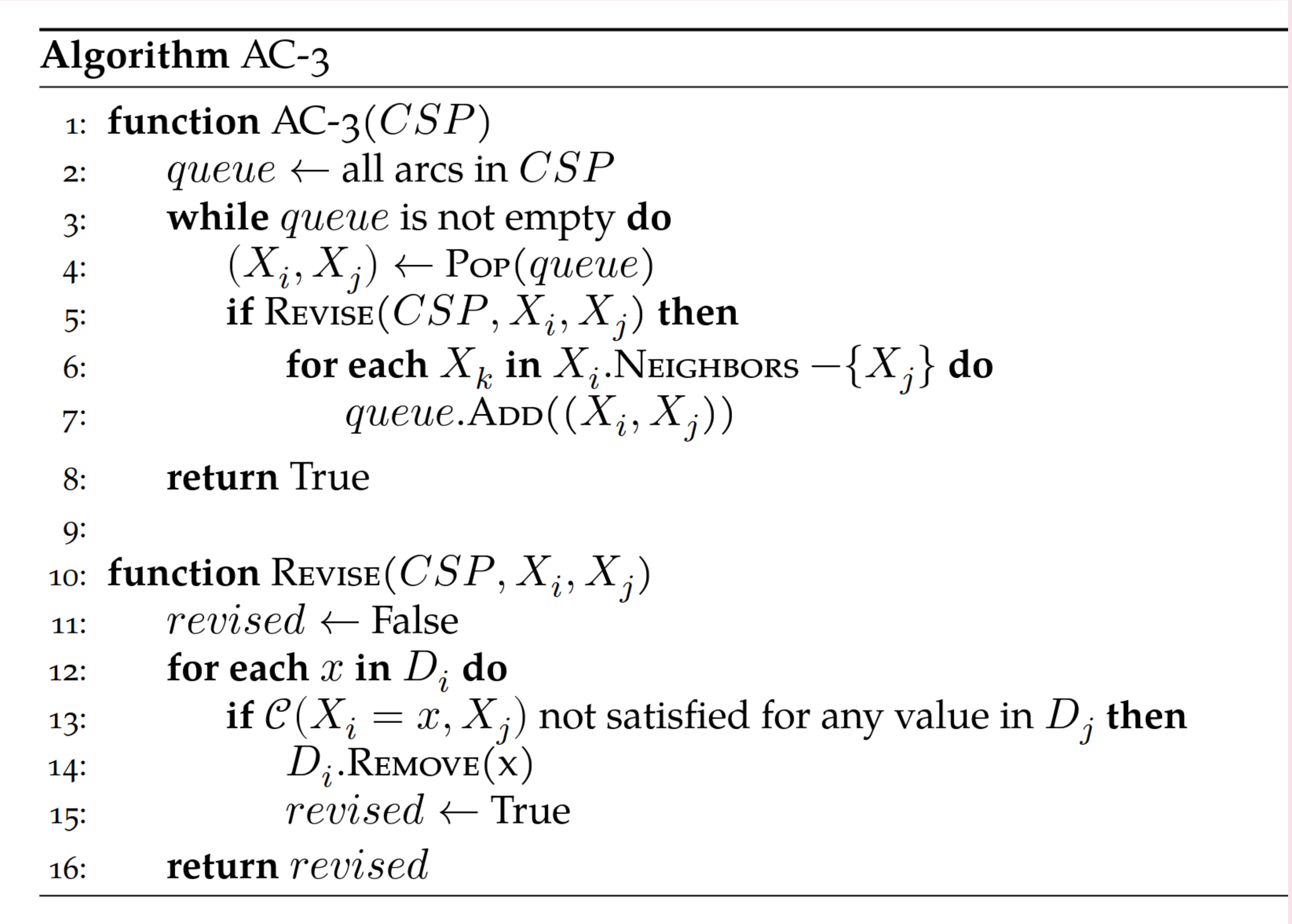
What Even Is Inference

- Constraints on one variable restrict others:
    - $X_1 \in \{A, B, C, D\}$ and $X_2 \in \{A\}$
    - $X_1 \neq X_2$
    - Inference: $X_1 \in \{B, C, D\}$
- If an unassigned variable has no domain…
    - Failure

Inference

- Arc consistency
    - Reduce domains for pairs of variables
- Path consistency
    - Assignment to two variables
    - Reduce domain of third variable

AC-3



Algorithm AC-3

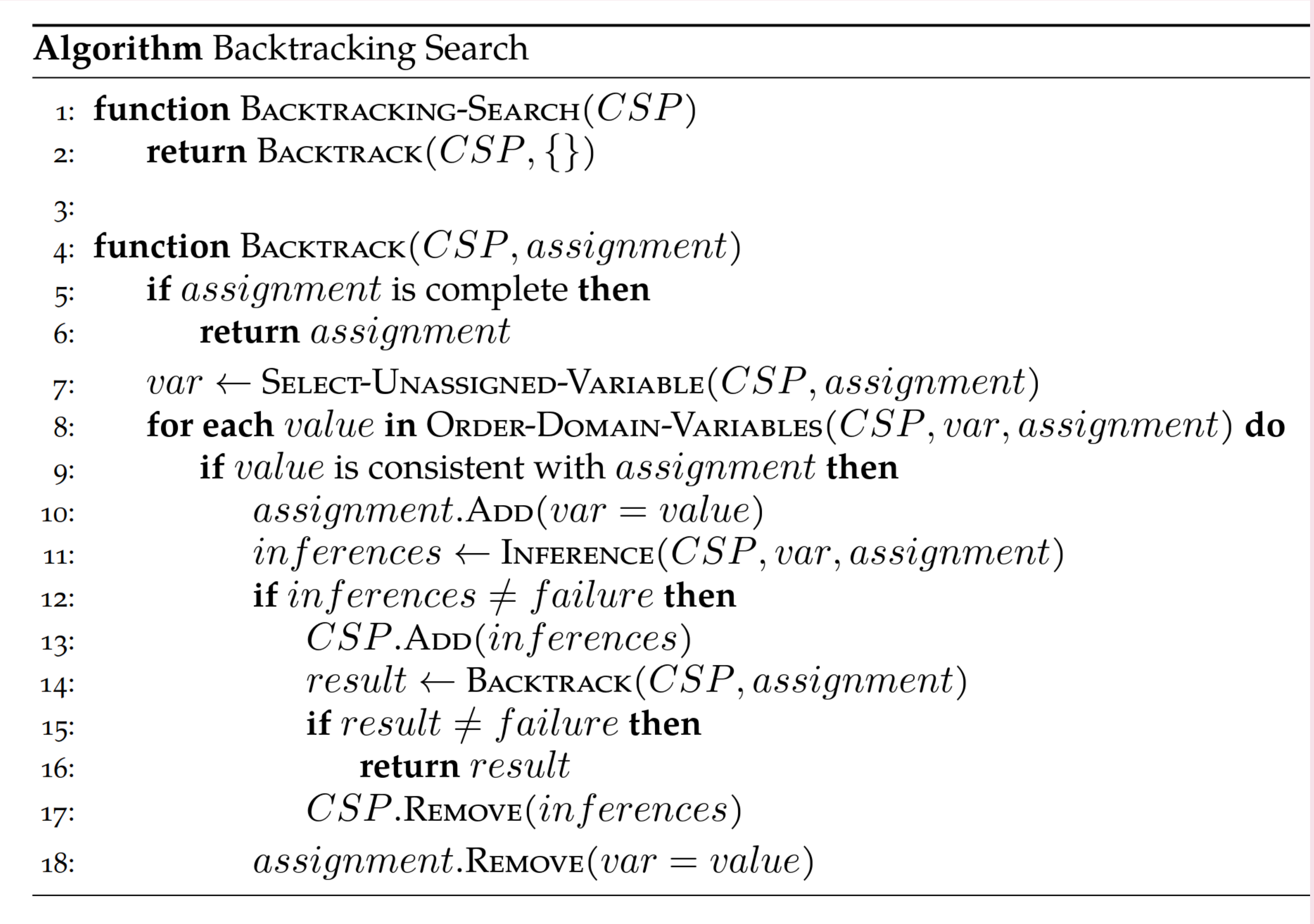
```
function AC-3($CSP$)
    $queue \leftarrow$ all arcs in $CSP$
    while $queue$ is not empty do
        $(X_i, X_j) \leftarrow$ Pop($queue$)
        if Revise($CSP, X_i, X_j$) then
            for each $X_k$ in $X_i$.Neighbors $-\{X_j\}$ do
                $queue$.Add($(X_i, X_j)$)
    return True

function Revise($CSP, X_i, X_j$)
    $revised \leftarrow$ False
    for each $x$ in $D_i$ do
        if $\mathcal{C}(X_i = x, X_j)$ not satisfied for any value in $D_j$ then
            $D_i$.Remove(x)
            $revised \leftarrow$ True
    return $revised$
```

How To Solve It (Again)

Backtracking search:

- Similar to DFS
- Variables are *ordered*
  - Why?
- Constraints checked each step
- Constraints optionally *propagated*

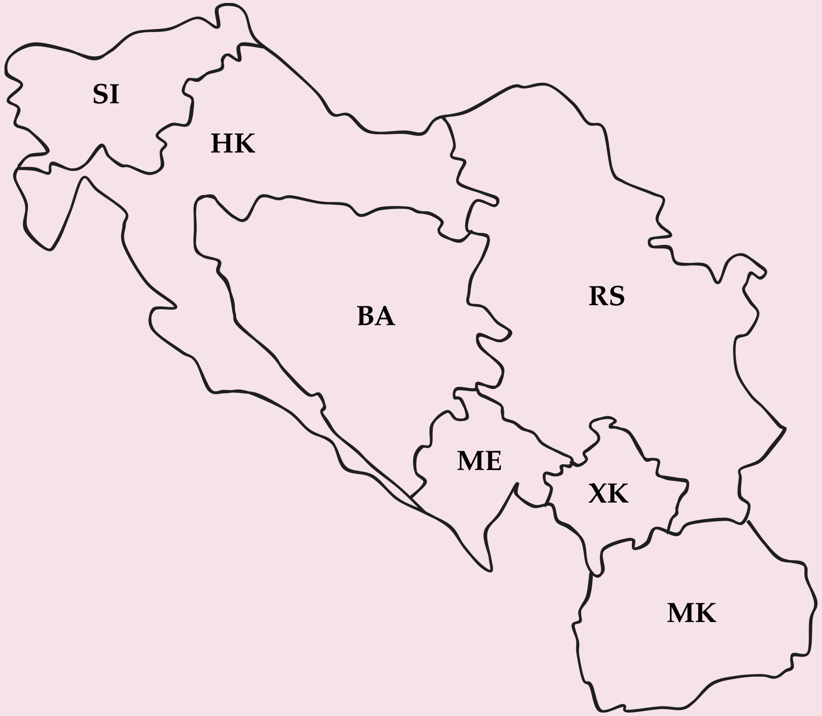
How To Solve It (Again)



Algorithm Backtracking Search

```
function BACKTRACKING-SEARCH($CSP$)
    return BACKTRACK($CSP, \{\}$)

function BACKTRACK($CSP, assignment$)
    if $assignment$ is complete then
        return $assignment$
    $var \leftarrow$ SELECT-UNASSIGNED-VARIABLE($CSP, assignment$)
    for each $value$ in ORDER-DOMAIN-VARIABLES($CSP, var, assignment$) do
        if $value$ is consistent with $assignment$ then
            $assignment$.ADD($var = value$)
            $inferences \leftarrow$ INFERENCE($CSP, var, assignment$)
            if $inferences \neq failure$ then
                $CSP$.ADD($inferences$)
                $result \leftarrow$ BACKTRACK($CSP, assignment$)
                if $result \neq failure$ then
                    return $result$
                $CSP$.REMOVE($inferences$)
            $assignment$.REMOVE($var = value$)
```

Yugoslav Arc Consistency



Ordering

- Select-Unassgined-Variable($CSP, assignment$)
  - Choose most-constrained variable[1]
- Order-Domain-Variables($CSP, var, assignment$)
  - Least-constraining value
- Why?

---

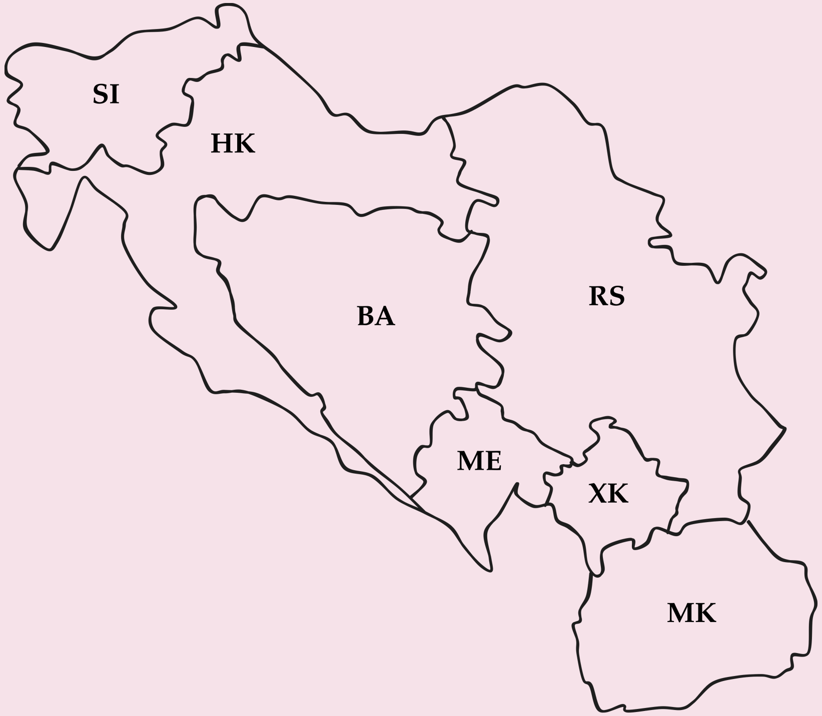
[1] or MRV: “Minimum Remaining Values”

Restructuring

Tree-structured CSPs:

- *Linear time* solution
- Directional arc consistency: $X_i \rightarrow X_{i+1}$
- Cutsets
- Sub-problems

Cutset Example



(Heuristic) Local Search

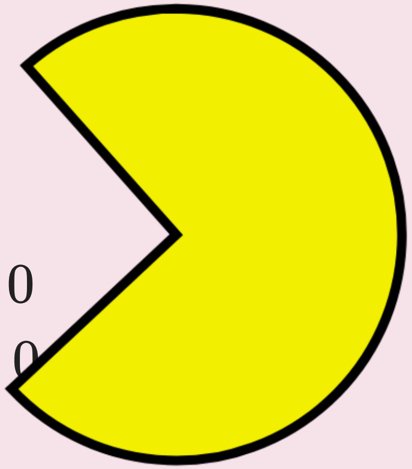
- Hill climbing
    - Random restarts
- Simulated annealing
- Fast?
- Complete?
- Optimal?

Continuous Domains

- Linear:

$$\begin{aligned} \max_{x} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \geq 0 \end{aligned}$$

- Convex



$$\begin{aligned} \min_{x} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0 \\ & h_i(\boldsymbol{x}) = 0 \end{aligned}$$

Is This Even Relevant in 2025?

- Absolutely yes.
- LLMs are bad at CSPs
- CSPs are common in the real world
    - Scheduling
    - Optimization
    - Dependency solvers

Logic

Yugoslav Logic

$R_{HK} \Rightarrow \neg R_{SI}$

$G_{HK} \Rightarrow \neg G_{SI}$

$B_{HK} \Rightarrow \neg B_{SI}$

$R_{HK} \vee G_{HK} \vee B_{HK}$

$\ldots$

Goal: find assignment of variables that satisfies conditions

Is It Possible To Know Things?

Yes.

😌

How Even Do We Know Things?

- What color is an apple?
    - Red?
    - Green?
    - Blue?
- Are you sure?

Symbols

- Propositional symbols
    - Similar to boolean variables
    - Either True or False

The Unambiguous Truth

- `It is a nice day.`
  - It is difficult to discern an unambiguous truth value.
- `It is warm outside.`
  - This has some truth value, but it is ambiguous.
- `The temperature is at least 78°F outside.`
  - This has an unambiguous truth value.[1]

[1] Provided that ‘outside’ is well-defined.

What Matters, Matters

- *Non-ambiguity* required
- Abitrary detail is not
- THE TEMPERATURE IS EXACTLY 78°F OUTSIDE.
    - We don’t necessarily need any other “related” symbols
- What is the problem?
- What do we care about?

Sentences

- What is a linguistic sentence?
    - Subject(s)
    - Verb(s)
    - Object(s)
    - *Relationships*
- What is a logical sentence?
    - Symbols
    - Relationships

Familiar Logical Operators

- $\neg$
    - "Not" operator, same as CS (`!`, `not`, etc.)
- $\wedge$
    - "And" operator, same as CS (`&&`, `and`, etc.)
    - This is sometimes called a *conjunction*.
- $\vee$
    - "Inclusive Or" operator, same as CS.
    - This is sometimes called a *disjunction*.

Unfamiliar Logical Operators

- $\Rightarrow$
  - Logical *implication*.
  - If $X_0 \Rightarrow X_1$, X_1 is always True when X_0 is True.
  - If X_0 is False, the value of X_1 is not constrained.
- $\iff$
  - “If and only If.”
  - If $X_0 \iff X_1$, X_0 and X_1 are either both True or both False.
  - Also called a *biconditional*.

Equivalent Statements

- $X_0 \Rightarrow X_1$ alternatively:
  - $(X_0 \wedge X_1) \vee \neg X_0$
- $X_0 \iff X_1$ alternatively:
  - $(X_0 \wedge X_1) \vee (\neg X_0 \wedge \neg X_1)$

- Can we make an XOR?

Knowledge Base & Queries

- We encode everything that we ‘know’
    - Statements that are true
- We query the knowledge base
    - Statement that we’d like to know about
- Logic:
    - Is statement consistent with KB?

Models

- Mathematical abstraction of problem
  - Allows us to solve it
- Logic:
  - Set of truth values for all sentences
  - …sentences comprised of symbols…
  - Set of truth values for all symbols
  - New sentences, symbols over time

Entailment

- $KB \models A$
  - "Knowledge Base entails A"
  - For every model in which KB is True, A is also True
  - One-way relationship: A can be True for models where KB is not True.
- Vocabulary: A is the *query*

Knowing Things

Falsehood:

- $KB \models \neg A$
  - No model exists where KB is True and A is True

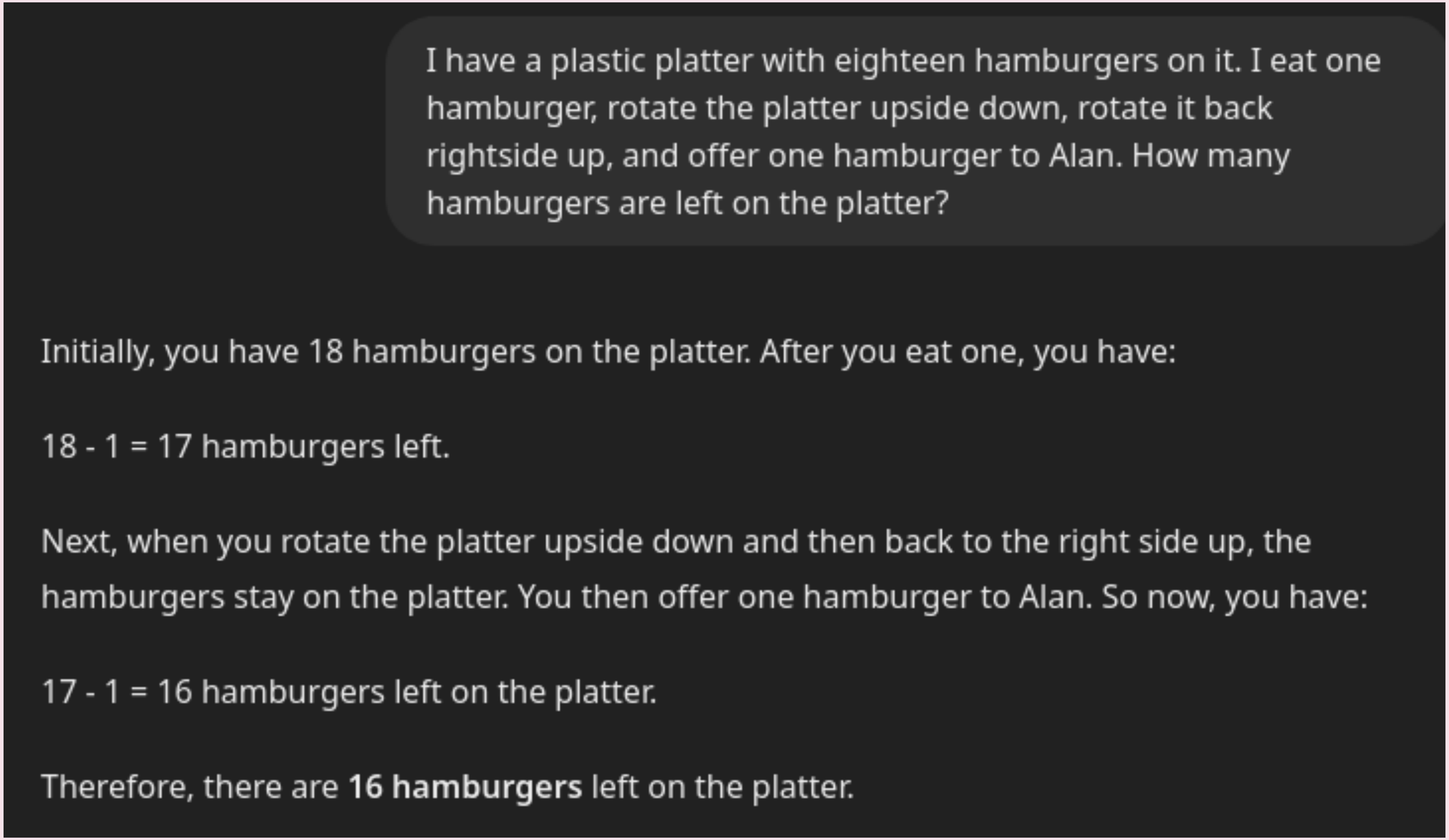
It is possible to not know things:[1]

- $KB \not\models A$
- $KB \not\models \neg A$

---

1. $\not\models$ – “does not entail”

It Is Possible To Not Know Things 😔



I have a plastic platter with eighteen hamburgers on it. I eat one hamburger, rotate the platter upside down, rotate it back rightside up, and offer one hamburger to Alan. How many hamburgers are left on the platter?

Initially, you have 18 hamburgers on the platter. After you eat one, you have:

18 - 1 = 17 hamburgers left.

Next, when you rotate the platter upside down and then back to the right side up, the hamburgers stay on the platter. You then offer one hamburger to Alan. So now, you have:

17 - 1 = 16 hamburgers left on the platter.

Therefore, there are **16 hamburgers** left on the platter.

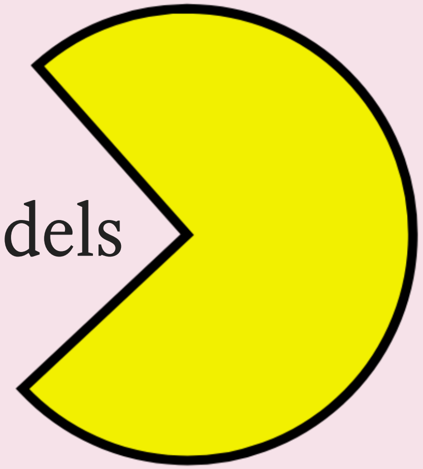
Lexicon

- *Valid*
  - $A \vee \neg A$
- *Satisfiable*
  - True for some models
- *Unsatisfiable*
  - $A \wedge \neg A$

Inference

- KB models real world
  - Truth values unambiguous
  - KB coded correctly
- $KB \models A$
  - A is true in the real world

Inference - How?



- Model checking
    - Enumerate possible models
    - We can do better
    - NP-complete 😣
- Theorem proving
    - Prove $KB \models A$

Satisfiability

- Commonly abbreviated “SAT”
    - Not the Scholastic Assessment Test
    - Much more difficult
    - *First* NP-complete problem
- The

Deliberate typographical error!

Satisfiability

- Commonly abbreviated “SAT”
- $(X_0 \wedge X_1) \vee X_2$
  - Satisfied by $X_0 = \text{True}$, $X_1 = \text{False}$, $X_2 = \text{True}$
  - Satisfied for any X_0 and X_1 if $X_2 = \text{True}$
- $X_0 \wedge \neg X_0 \wedge X_1$
  - Cannot be satisfied by any values of X_0 and X_1

Satisfaction

- SAT reminiscent of Constraint Satisfaction Problems
- CSPs reduce to SAT
  - Solving SAT → solving CSPs
  - Restricted to specific operators
  - CSP global constraints → refactor as binary
- Still NP-Complete

Why Do I Keep On Doing This To You

This is the entire point of the course.

Theory and practice are the same, in theory, but in practice they differ.

CSP Solution Methods

- They all work
- Backtracking search
- Hill-climbing
- Ordering (?)

SAT Solvers

- Heuristics
- PicoSAT
    - Python bindings: `pycosat`
    - (Solver written in C) (it’s fast)
- You don’t have to know anything about the problem
    - This is not actually true
- Conjunctive Normal Form

Conjunctive Normal Form

- *Literals* — symbols or negated symbols
  - X_0 is a literal
  - $\neg X_0$ is a literal
- *Clauses* — combine literals and disjunction using disjunctions ($\vee$)
  - $X_0 \vee \neg X_1$ is a valid disjunction
  - $(X_0 \vee \neg X_1) \vee X_2$ is a valid disjunction

Conjunctive Normal Form

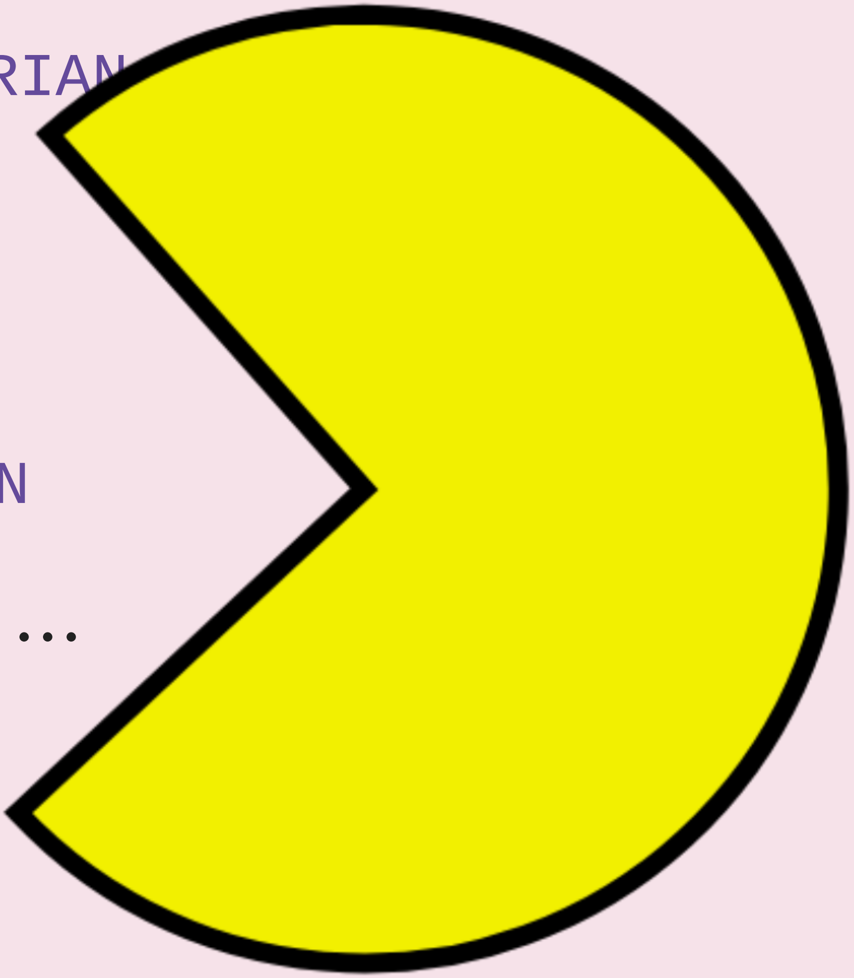
- *Conjunctions* ($\wedge$) combine clauses (and literals)
  - $X_1 \wedge (X_0 \vee \neg X_2)$
- Disjunctions cannot contain conjunctions:
- $X_0 \vee (X_1 \wedge X_2)$ not in CNF
  - Can be rewritten in CNF: $(X_0 \vee X_1) \wedge (X_0 \vee X_2)$

Converting to CNF

- $X_0 \iff X_1$
    - $(X_0 \Rightarrow X_1) \wedge (X_1 \Rightarrow X_0)$
- $X_0 \Rightarrow X_1$
    - $\neg X_0 \vee X_1$
- $\neg(X_0 \wedge X_1)$
    - $\neg X_0 \vee \neg X_1$
- $\neg(X_0 \vee X_1)$
    - $\neg X_0 \wedge \neg X_1$

Limitations

- Consider: No cat is a vegetarian
- Express in propositional symbols?
- ¬ First cat is a vegetarian
- ¬ Second cat is a vegetarian
- ¬ Third cat is a vegetarian …



Solutions

First-Order Logic:

- ∀ (“for all”)
- ∃ (“there exists at least one”)

Loops 🙂 :

```
for cat in cats:
  t = Expr(f"{cat} is not a vegetarian")
  Exprs.push(t)
```

Goal: find assignment of variables that satisifies conditions

References

- Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* 4th Edition, 2020.
- Mykal Kochenderfer, Tim Wheeler, and Kyle Wray. *Algorithms for Decision Making.* 1st Edition, 2022.
- Stanford CS231
- Stanford CS228
- UC Berkeley CS188