6.034 Artificial Intelligence: Lecture 6
Constraint Satisfaction

Professor Robert C. Berwick

September 16, 2020

Rules

- Lines in a drawing can meet up in a few different ways (these are the “constraints”)
- Places where lines meet up are called junctions
- Not all junctions are physically realizable

Requirements

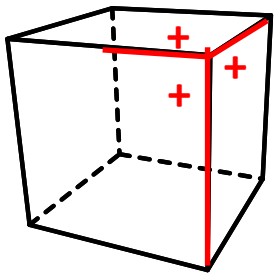
- Three assumptions
- Vocabulary of Junction types

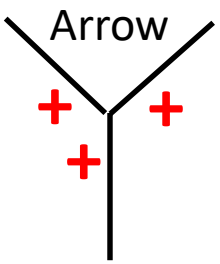
Three Assumptions

1) General position
   - no “screw cases”
2) Trihedral vertices
   - all line junctions formed by intersection of 3 planes
3) Four line labels

Note: also by convention, we assume boundaries are traversed “clockwise”

Vocabulary of Junction types

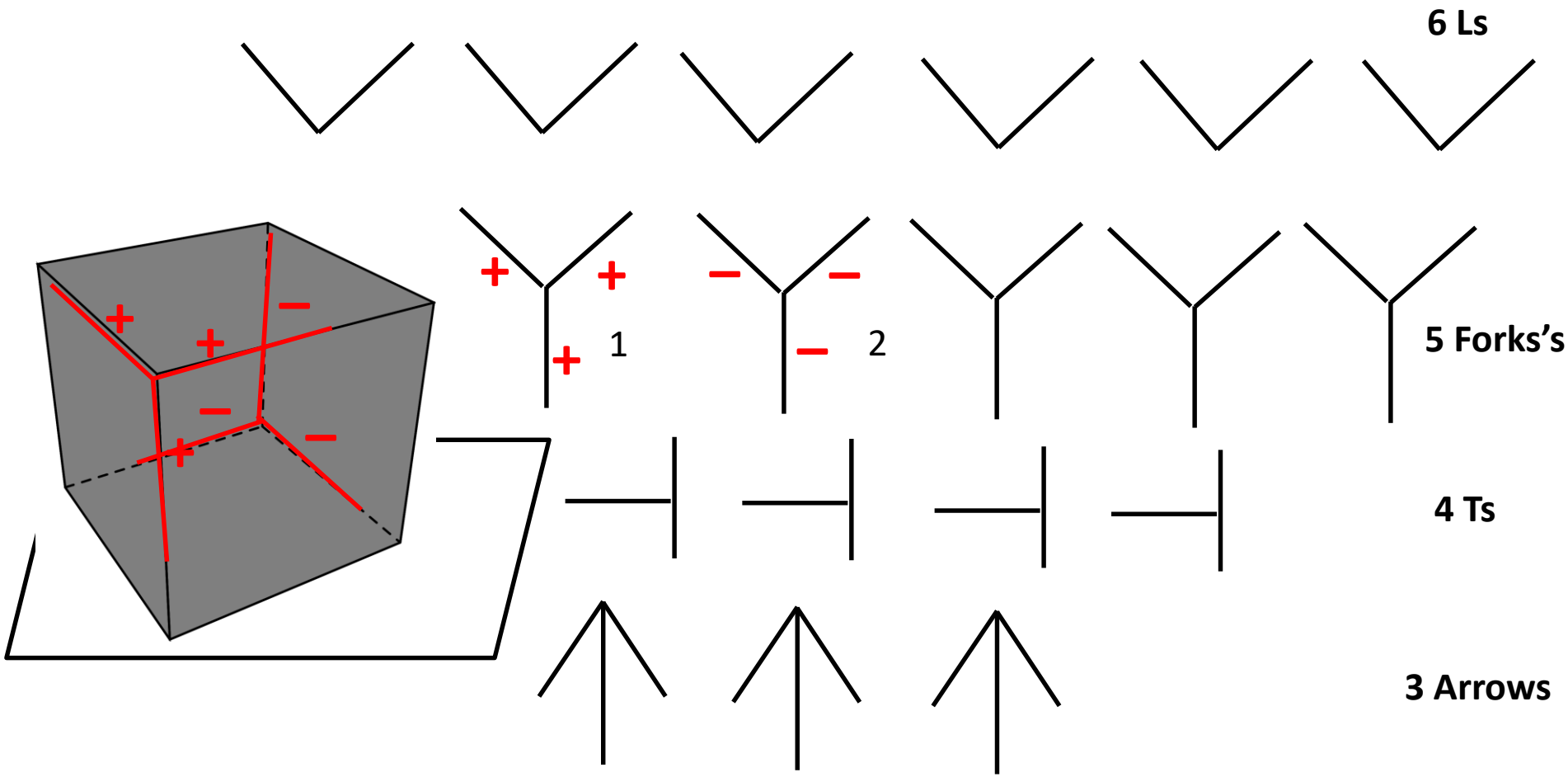




18 atomic <u>junction</u> (vertex) types
(where 2 or 3 lines meet), assuming 3-faced vertices:

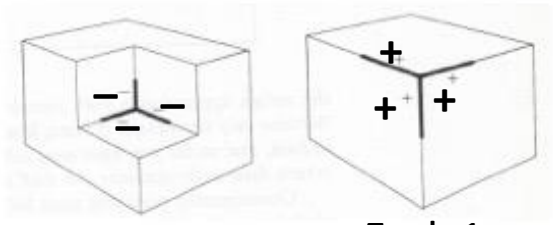
(i) 6 <u>L</u>s
(ii) 5 <u>F</u>orks
(iii) 4 <u>T</u>s
(iv) 3 <u>A</u>rrows

Vocabulary of junction types

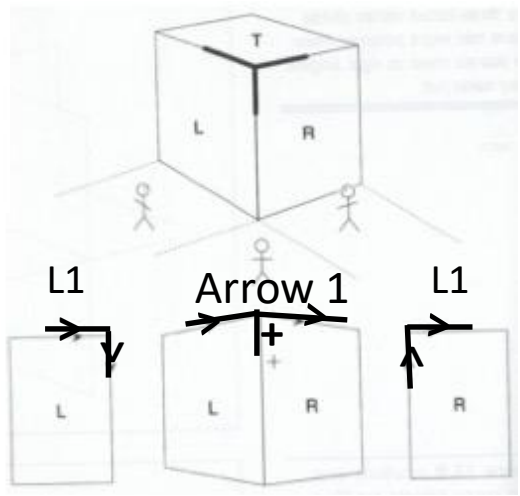


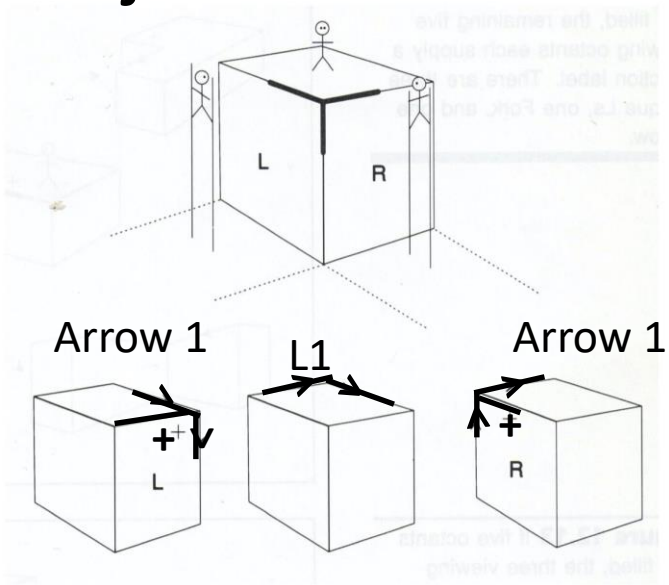
These come from different views of the object

The full library of junction types – just 18 labels

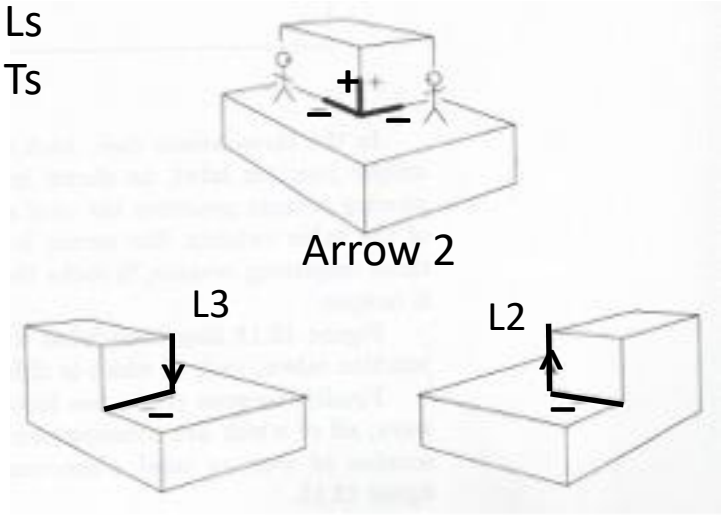


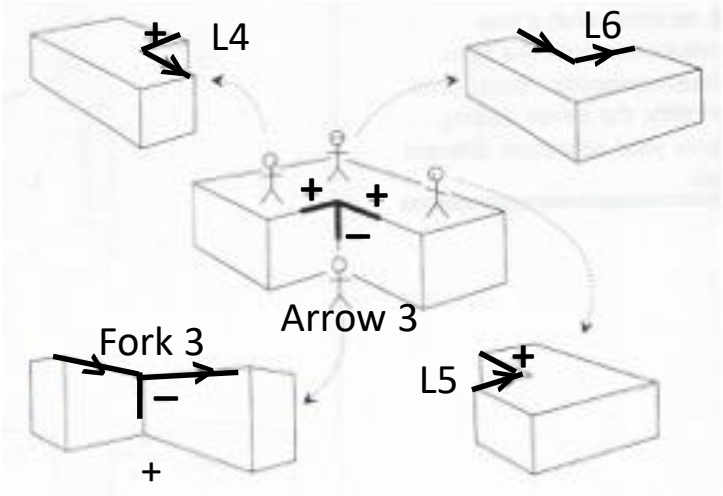
Forks



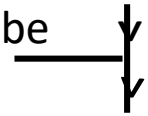


3 Arrows
5 Forks
6 Ls
4 Ts





And 4 T's, since the "T" is always a boundary line, and the other line hitting the "T" can be any of the other 4 labels, +, –, >, or >



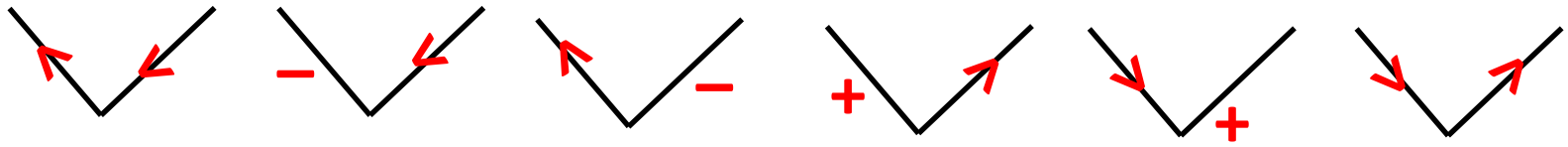
+ Rotations = Forks 4, 5

The "periodic table" of 18 "elements" that can label trihedral junctions

6 L's

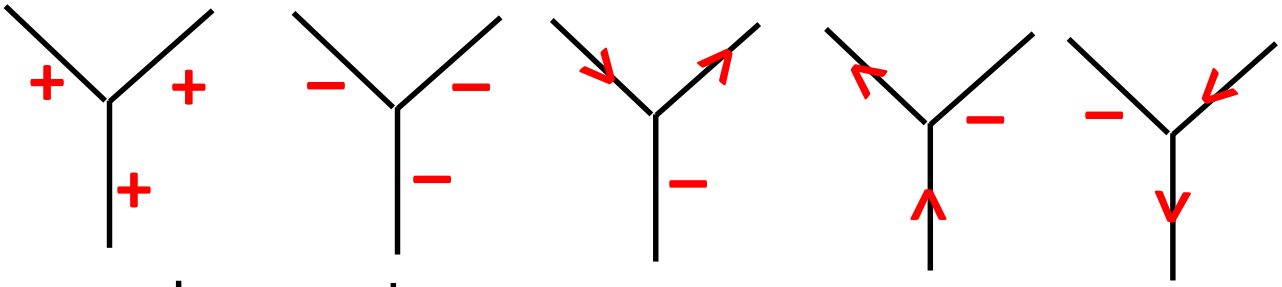
L's

4^2 =16



Forks

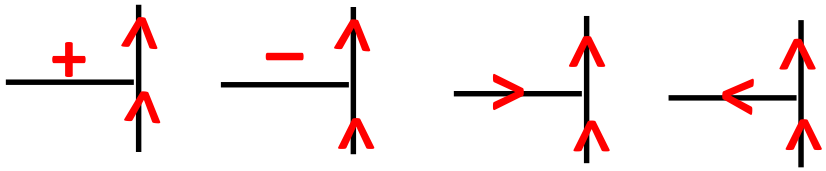
4^3 = 64



5 F's

T's

4^3 =64

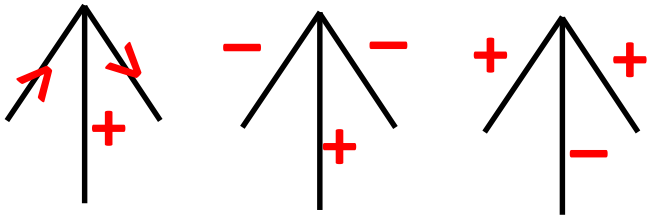


4 T's

3 A's

Arrows

4^3 = 64



Arrow 1 Arrow 2 Arrow 3

Q: How many junction types <u>logically</u> <u>possible</u> if no real world 3-vertex constraint?

Total: 208 **18 physically realizable out of 208 logically possible – constraints**

Now let's see how to do "constraint satisfaction" using these constraints

Vocabulary for a general method: the Domain Reduction

- Variable V: something that can have an assignment
- Value x: something that can be assigned
- Domain D: a bag of values
- Constraint C: a condition that must be satisfied among variable values

Systematic Idea for Map Coloring: *Domain Reduction Algorithm*

- For each depth first search assignment
  - For each variable V_i <u>*considered*</u>

    we have choices here
    - For each value x_i in D_i (domain of V_i)
      - For each constraint C between V_i and other variables V_j

        we use binary constraints (e.g., Y/N)
        - If $\nexists x_j \in D_j$ such that $C(x_i, x_j)$ is satisfied
        - Then remove x_i from D_i
        - If D_i empty, then backtrack

Domain Reduction Algorithm

Summary: Constraint satisfaction

- Different architectures/algorithms that exploit
  - Constraints exposed by
    - Representations that support
      - Models of perception, thinking, and action

Many different, difficult problems can be solved this way!