Randomized Model Finder

Difficult to solve ?

Let’s try to guess…

Model Finding Basics

- First Order Logic Formula
  - Predicate
  - Functions
- Interpretation
  - (Finite) Domain
  - Interpretation of predicates and functions
- Model: Interpretation that satisfies some FOL formulas

Why model finding ?

- Artificial Intelligence
- Constraint satisfaction problems

- Disproof a formula
- Show non respect of a specification

How to find model ?

- Exhaustive search
- SEM: Search using constraint propagation method
- MACE: Translating « instanciated » FOL formulas into propositional clauses, solved by a SAT-Solver
- KODKOD: Takes into account partial instance

TPTP

- A language to write FOL formulas and propositional clauses
- Annotations
  - Kind of formula (conjecture, axioms)
  - Name
- Huge library spanning across several domains used to test and compare automatic reasoning tools.

TPTP

- Formulas:
  - F := **F** & **F** | ![*x*]. **F** | ?[*x*].**F** | ... | **A**
  - A := Predicate($\mathbf{T}_1$,...,$\mathbf{T}_n$)
  - T := Symbol($\mathbf{T}_1$,...,$\mathbf{T}_n$)
- Example:
  ![H1,H2] :(q(H1,H2)<=>H1 = H2)

Solver (naive)

1. Pick an interpretation *I*
2. Evaluate *I* on input formulas
3. If *I* satisfies all formulas → output(*I*)
4. If the max number of iterations has been reached → terminates
5. Go to (1)

Why randomize it ?

- Finding models takes time...
- ... and space
- Exponential complexity makes enumeration impossible

- Possibility to find a model but not to proove their inexistance

Improve your chances...

- Since we can't look at all the interpretations, we have to select the most promising ones.
- Selection based on a cost function and a heuristic to navigate through the search space

How cheap is your interpretation ?

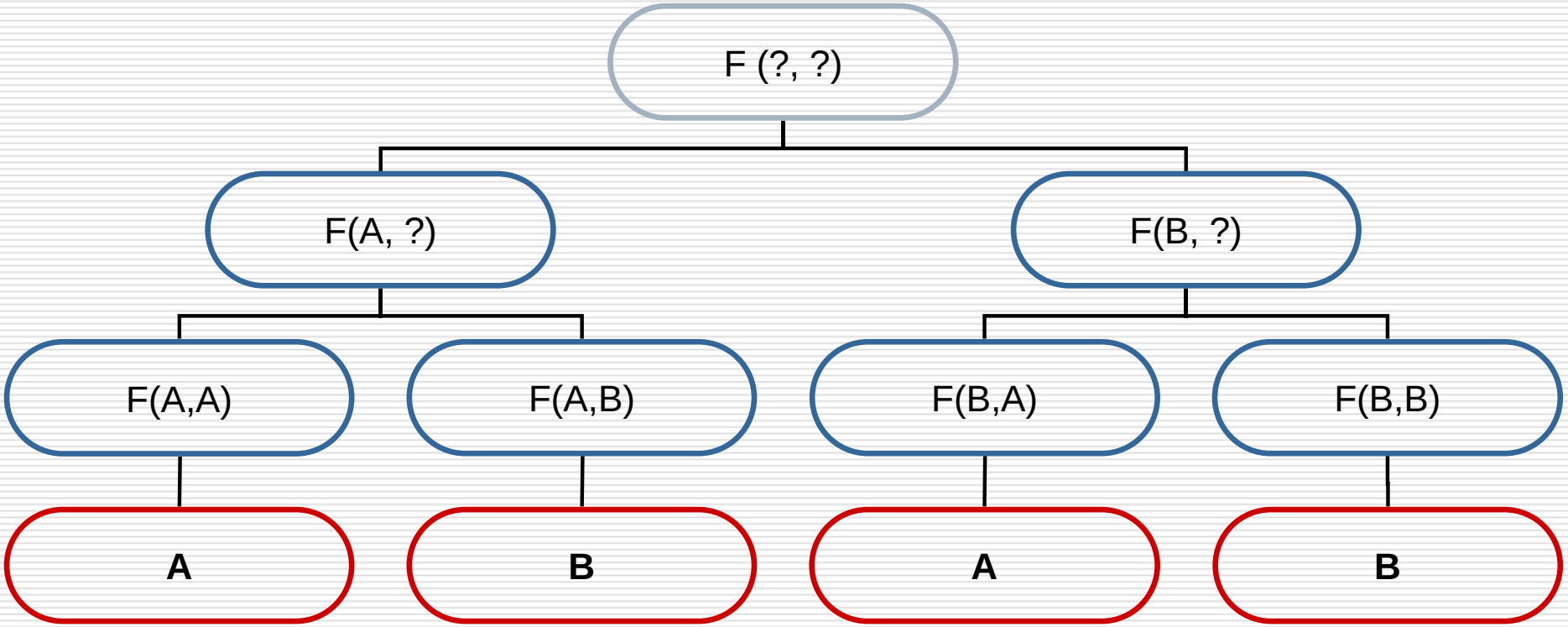
- Depth of first « broken » atom
- Number of « broken » atoms

- Idea: if the cost of *I* is zero, then your formula is satisfied.

- « Reduce » model finding to optimizing a function over an high dimension discrete search space

Interpretation

- Function interpretation is encoded as a list of Elements
- Predicate interpretation is encoded as a list of Boolean
- Interpretation: encoded as a vector



Navigating in the search space

- Particle Swarm Optimization
  - Particles moving around with some initial speed. The minima they find is used to generate speeds for the next iteration.
- Local search