

Logical Inference 1

introduction

Chapter 9

Some material adopted from notes by Andreas Geyer-Schulz,, Chuck Dyer, and Mary Getoor

Overview

- A: Model checking for propositional logic
- Rule based reasoning in first-order logic
 - Inference rules and generalized modes ponens
 - -Forward chaining
 - -Backward chaining
- Resolution-based reasoning in first-order logic
 - -Clausal form
 - Unification
 - -Resolution as search
- Inference wrap up

Propositional Logic Model checking

- Given KB, does sentence S hold?
- Basically generate and test:
 - -Generate all possible models
 - -Consider models M in which KB is TRUE
 - $-If \forall M S$, then S is **provably true**
 - $-If \forall M \neg S$, then S is **provably false**
 - -Otherwise ($\exists M1 \ S \land \exists M2 \neg S$): S is **satisfiable** but neither provably true or provably false

From Satisfiability to Proof (1)

- To see if a satisfiable KB entails sentence S, see if KB ∧ ¬S is satisfiable
 - -If it is not, then the KB entails S
 - -If it is, then the KB does not email S
 - -This is a refutation proof
- Consider the KB with (P, P=>Q, ~P=>R)
 Does the KB it entail Q? R?

Efficient PL model checking (1)

<u>Davis-Putnam algorithm</u> (DPLL) is <u>generate-and-</u> <u>test</u> model checking with several optimizations:

- *Early termination:* short-circuiting of disjunction/ conjunction
- *Pure symbol heuristic*: symbols appearing only negated or unnegated must be FALSE/TRUE respectively
 - e.g., in $[(A \lor \neg B), (\neg B \lor \neg C), (C \lor A)] \land \& B$ are pure, C impure. Make pure symbol literal true: if there's a model for S, making pure symbol true is also a model
- Unit clause heuristic: Symbols in a clause by itself can immediately be set to TRUE or FALSE

Using the AIMA Code

python> python

Python ...

>>> from logic import *

>>> expr('P & P==>Q & ~P==>R')

((P & (P >> Q)) & (~P >> R))

>>> dpll_satisfiable(expr('P & P==>Q & ~P==>R'))
{R: True, P: True, Q: True}

>>> dpll_satisfiable(expr('P & P==>Q & ~P==>R & ~R'))
{R: False, P: True, Q: True}

>>> dpll_satisfiable(expr('P & P==>Q & ~P==>R & ~Q')) False

The KB entails Q but does not email R

expr parses a string, and

dpll satisfiable returns a

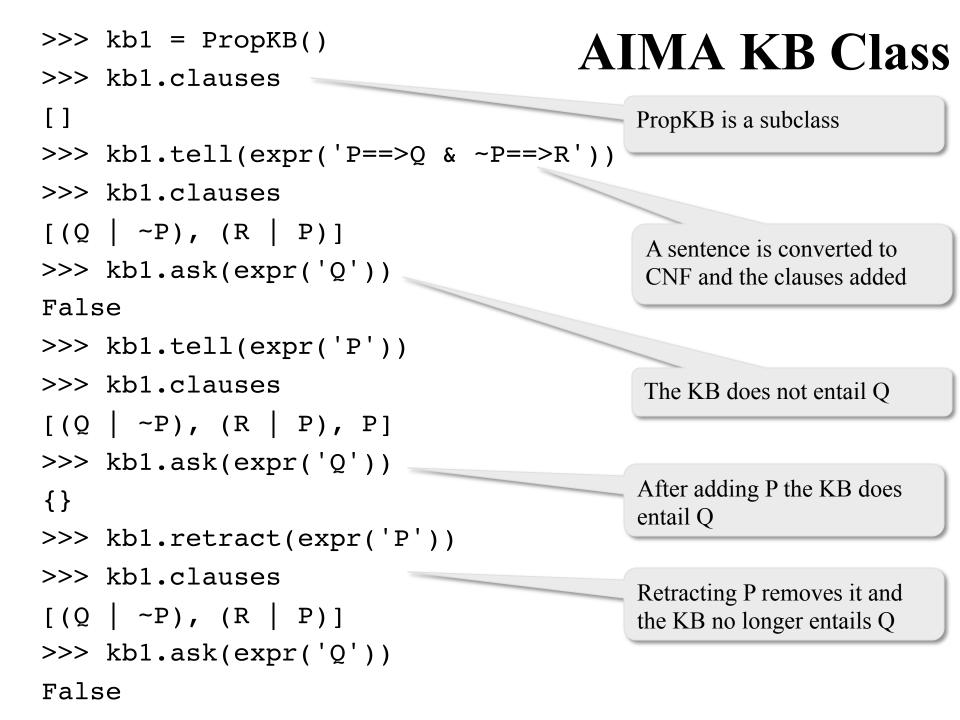
model if satisfiable else False

returns a logical expression

>>>

Efficient PL model checking (2)

- <u>WalkSAT</u> is a local search for satisfiability: Pick a symbol to flip (toggle TRUE/FALSE), either using min-conflicts *or* choosing randomly
- ... or you can use *any* local or global search algorithm!
- There are many model checking algorithms and systems
 - -See for example, MiniSat
 - –<u>International SAT Competition</u> (2003...2016)



Reminder: Inference rules for FOL

- Inference rules for propositional logic apply to FOL as well
 - Modus Ponens, And-Introduction, And-Elimination, ...
- New (sound) inference rules for use with quantifiers:
 - Universal elimination
 - -Existential introduction
 - -Existential elimination
 - Generalized Modus Ponens (GMP)