CSCI 5582 Artificial Intelligence

Lecture 11

Jim Martin

Today 10/3

- Review Model Checking/Wumpus
- CNF
- WalkSat
- Break
- Start on FOL

Review

- Propositional logic provides
  - Propositions that have
  - Truth values and
  - Logical connectives that allow a
  - Compositional Semantics and
  - Inference

Models

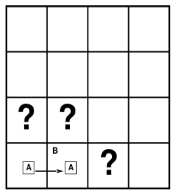
- Models are formally structured worlds with respect to which truth can be evaluated.
- *m* is a model of a sentence α if α is true in *m*
- *M(α)* is the set of all models of α

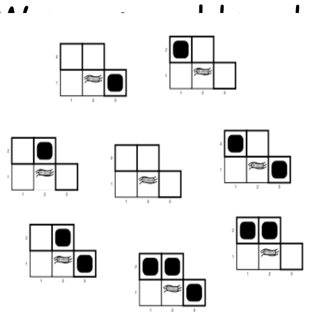
Wumpus world model

Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

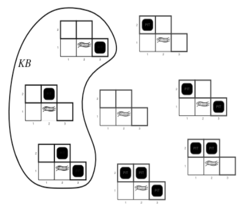
Consider possible models for ?s assuming only pits

3 Boolean choices $\Rightarrow$ 8 possible models





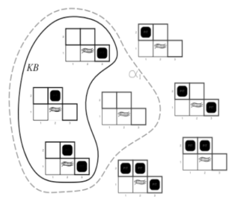
Wumpus world model



KB = wumpus-world rules + observations

CSCI 5582 Fall 2006

Wumpus world model

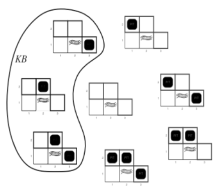


KB = wumpus-world rules + observations

α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking

CSCI 5582 Fall 2006

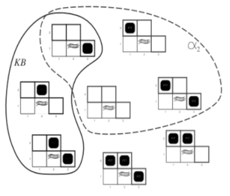
Wumpus world model



KB = wumpus-world rules + observations

CSCI 5582 Fall 2006

Wumpus world model



KB = wumpus-world rules + observations

α_2 = "[2,2] is safe", $KB \not\models \alpha_2$

Effective propositional inference

- Two families of efficient algorithms for propositional inference based on model checking:
- Are used for checking satisfiability
- Complete backtracking search algorithms
  - DPLL algorithm (Davis, Putnam, Logemann, Loveland)
  - Incomplete local search algorithms
    - `WalkSAT` algorithm

Conversion to CNF

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

- Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
  - $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$.
  - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
- Move $\neg$ inwards using de Morgan's rules and double-negation:
  - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
- Apply distributivity law ($\wedge$ over $\vee$) and flatten:
  - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

The DPLL algorithm

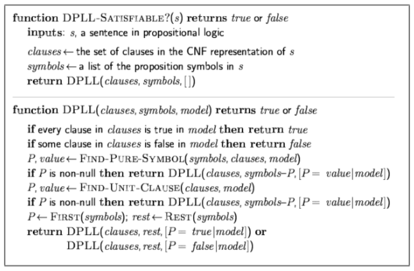
- Determine if an input propositional logic sentence (in CNF) is satisfiable by assigning values to variables.

1. Pure symbol heuristic
   Pure symbol: always appears with the same "sign" in all clauses.
   e.g., In the three clauses (A ∨ ¬B), (¬B ∨ ¬C), (C ∨ A), A and B are pure, C is impure.
   Assign a pure symbol so that their literals are true.

2. Unit clause heuristic
   Unit clause: only one literal in the clause or only one literal which has not yet received a value. The only literal in a unit clause must be true.

CSCI 5582 Fall 2006

The DPLL algorithm



```
function DPLL-SATISFIABLE?(s) returns true or false
    inputs: s, a sentence in propositional logic

    clauses ← the set of clauses in the CNF representation of s
    symbols ← a list of the proposition symbols in s
    return DPLL(clauses, symbols, [])

function DPLL(clauses, symbols, model) returns true or false

    if every clause in clauses is true in model then return true
    if some clause in clauses is false in model then return false
    P, value ← FIND-PURE-SYMBOL(symbols, clauses, model)
    if P is non-null then return DPLL(clauses, symbols–P, [P = value|model])
    P, value ← FIND-UNIT-CLAUSE(clauses, model)
    if P is non-null then return DPLL(clauses, symbols–P, [P = value|model])
    P ← FIRST(symbols); rest ← REST(symbols)
    return DPLL(clauses, rest, [P = true|model]) or
           DPLL(clauses, rest, [P = false|model])
```

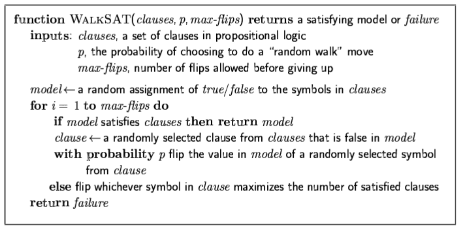
CSCI 5582 Fall 2006

The `WalkSAT` algorithm

- Incomplete, local search algorithm.
- Evaluation function: The min-conflict heuristic of minimizing the number of unsatisfied clauses.
- Steps are taken in the space of complete assignments, flipping the truth value of one variable at a time.
- Balance between greediness and randomness.
  - To avoid local minima

CSCI 5582 Fall 2006

The WalkSAT algorithm



```
function WALKSAT(clauses, p, max-flips) returns a satisfying model or failure
   inputs: clauses, a set of clauses in propositional logic
           p, the probability of choosing to do a "random walk" move
           max-flips, number of flips allowed before giving up

   model ← a random assignment of true/false to the symbols in clauses
   for $i = 1$ to max-flips do
       if model satisfies clauses then return model
       clause ← a randomly selected clause from clauses that is false in model
       with probability p flip the value in model of a randomly selected symbol
           from clause
       else flip whichever symbol in clause maximizes the number of satisfied clauses
   return failure
```

Break

- Quiz 1: Average was 43

Pros and cons of propositional logic

☺ Propositional logic is declarative
☺ Propositional logic allows partial/disjunctive/negated information
  – (unlike most data structures and databases)
☺ Propositional logic is compositional:
  – meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
☺ Meaning in propositional logic is context-independent
  – (unlike natural language, where meaning depends on context)
☹ Propositional logic has very limited expressive power
  – (unlike natural language)
  – E.g., cannot say "pits cause breezes in adjacent squares"
    • except by writing one sentence for each square

FOL

- At a high level…
  - FOL allows you to represent objects, properties of objects, and relations among objects
  - Specific domains are modeled by developing knowledge-bases that capture the important parts of the domain (change, auto repair, medicine, time, set theory, etc)

FOL

- First order logic adds
  - Variables and quantifiers that allow
  - Statements about unknown objects and
  - Statements about classes of objects

First-order logic

- Whereas propositional logic assumes the world contains facts,
- first-order logic (like natural language) assumes the world contains
  - Objects: people, houses, numbers, colors, baseball games, wars, …
  - Relations: red, round, prime, brother of, bigger than, part of, comes between, …
  - Functions: father of, best friend, one more than, plus, …

Syntax of FOL

Constants	KingJohn, 2, ,...
Predicates	Brother, >,...
Functions	Sqrt, LeftLegOf,...
Variables	x, y, a, b,...
Connectives	¬, ⇒, ∧, ∨, ⇔
Equality	=
Quantifiers	∀, ∃

Atomic sentences

Atomic sentence = *predicate* (*term*$_1$,...,*term*$_n$)
or *term*$_1$ = *term*$_2$

Term = *function* (*term*$_1$,...,*term*$_n$)
or *constant* or *variable*

- E.g.,
  - *Brother(KingJohn, RichardTheLionheart)*
  - *> (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))*

Complex sentences

- Complex sentences are made from atomic sentences using connectives

$\neg S$, $S_1 \wedge S_2$, $S_1 \vee S_2$, $S_1 \Rightarrow S_2$, $S_1 \Leftrightarrow S_2$,

E.g.
Sibling(KingJohn,Richard) ⇒
Sibling(Richard,KingJohn)

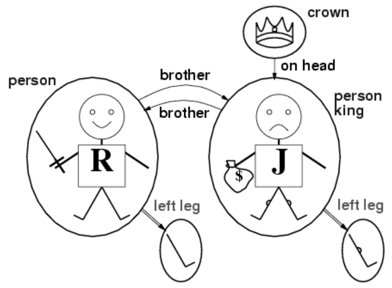
Truth in first-order logic

- Sentences are true with respect to a model and an interpretation
- Model contains objects (domain elements) and relations among them
- Interpretation specifies referents for

constant symbols	→	objects
predicate symbols	→	relations
function symbols	→	functional relations

- An atomic sentence *predicate*($term_1,\ldots,term_n$) is true iff the objects referred to by $term_1,\ldots,term_n$ are in the relation referred to by *predicate*.

Models for FOL: Example



Models as Sets

- Let's populate a domain:
  - {R, J, RLL, JLL, C}
- Property Predicates
  - Person = {R, J}
  - Crown = {C}
  - King = {J}
- Relational Predicates
  - Brother = { <R,J>, <J,R>}
  - OnHead = {<C,J>}
- Functional Predicates
  - LeftLeg = {<R, RLL>, <J, JLL>}

Quantifiers

- Allows us to express properties of collections of objects instead of enumerating objects by name
- Universal: "for all" ∀
- Existential: "there exists" ∃

Universal quantification

∀*<variables> <sentence>*

Everyone at CU is smart:
$\forall x\ At(x, CU) \Rightarrow Smart(x)$

∀x *P* is true in a model *m* iff *P* is true with *x* being each possible object in the model

Roughly speaking, equivalent to the conjunction of instantiations of *P*

$At(KingJohn,CU) \Rightarrow Smart(KingJohn)$
$\wedge\ At(Richard,CU) \Rightarrow Smart(Richard)$
$\wedge\ At(Ralphie,CU) \Rightarrow Smart(Ralphie)$
$\wedge\ \ldots$

Existential quantification

∃*<variables> <sentence>*

Someone at CU is smart:
$\exists x\ At(x, CU) \wedge Smart(x)$

∃*x P* is true in a model *m* iff *P* is true with *x* being some possible object in the model

- Roughly speaking, equivalent to the disjunction of instantiations of *P*

$At(KingJohn,CU) \wedge Smart(KingJohn)$
$\vee\ At(Richard,CU) \wedge Smart(Richard)$
$\vee\ At(Ralphie, CU) \wedge Smart(VUB)$
$\vee\ \ldots$

Properties of quantifiers

∀x ∀y is the same as ∀y ∀x
∃x ∃y is the same as ∃y ∃x

∃x ∀y is not the same as ∀y ∃x
∃x ∀y Loves(x,y)
- "There is a person who loves everyone in the world"

∀y ∃x Loves(x,y)
- "Everyone in the world is loved by at least one person"

- Quantifier duality: each can be expressed using the other
  ∀x Likes(x,IceCream)   ¬∃x ¬Likes(x,IceCream)
  ∃x Likes(x,Broccoli)   ¬∀x ¬Likes(x,Broccoli)

CSCI 5582 Fall 2006

Variables

- A big part of using FOL involves keeping track of all the variables while reasoning.
- Substitution lists are the means used to track the value, or binding, of variables as processing proceeds.

[illegible]

CSCI 5582 Fall 2006

Examples

[illegible]

[illegible]

[illegible]

[illegible]

CSCI 5582 Fall 2006

Examples

[illegible]
[illegible]
[illegible]

CSCI 5582 Fall 2006

Inference

- Inference in FOL involves showing that some sentence is true, given a current knowledge-base, by exploiting the semantics of FOL to create a new knowledge-base that contains the sentence in which we are interested.

CSCI 5582 Fall 2006

Inference Methods

- Proof as Generic Search
- Proof by Modus Ponens
  - Forward Chaining
  - Backward Chaining
- Resolution
- Model Checking

CSCI 5582 Fall 2006

Generic Search

- States are snapshots of the KB
- Operators are the rules of inference
- Goal test is finding the sentence you're seeking
  - I.e. Goal states are KBs that contain the sentence (or sentences) you're seeking

CSCI 5582 Fall 2006

Example

- Harry is a hare [illegible]
- Tom is a tortoise [illegible]
- Hares outrun tortoises

  [illegible]
- Harry outruns Tom?

CSCI 5582 Fall 2006

Tom and Harry

- And introduction

  [illegible]
- Universal elimination

  [illegible]
- Modus ponens

  [illegible]

CSCI 5582 Fall 2006

What's wrong?

- The branching factor caused by the number of operators is huge
- It's a blind (undirected) search

So…

- So a reasonable method needs to control the branching factor and find a way to guide the search…
- Focus on the first one first

Forward Chaining

- When a new fact *p* is added to the KB
  - For each rule such that p unifies with part of the premise
    - If all the other premises are known
    - Then add consequent to the KB

This is a data-driven method.

Backward Chaining

- When a query q is asked
  - If a matching q' is found return substitution list
  - Else For each rule q' whose consequent matches q, attempt to prove each antecedent by backward chaining

This is a goal-directed method. And it's the basis for Prolog.

Backward Chaining

[redacted]

Is Tom faster than someone?

Notes

- Backward chaining is not abduction; we are not inferring antecedents from consequents.
- The fact that you can't prove something by these methods doesn't mean its false. It just means you can't prove it.

Resolution

- Modus ponens is not complete. I.e. there are things we should be able to prove true that we can't by using Modus ponens alone.
- Used appropriately, resolution is complete.

CSCI 5582 Fall 2006

Resolution Example

[illegible]

[illegible]

[illegible]

[illegible]

CSCI 5582 Fall 2006

Resolution Example

Convert to Normal Form

[illegible]

[illegible]

[illegible]

[illegible]

Resolve 1 and 3

[illegible]

Resolve 2 and 5

[illegible]

Resolve 4 and 6

[illegible]

CSCI 5582 Fall 2006