Foundations of Artificial Intelligence

CS472/3 — Fall 1999
Lecture #20

Bart Selman

Today's Lecture

Inference
Resolution / Unification
Chapter 8 & 9, R&N.

Last time: Formal Representation of Circuit

- circuit diagram captured in

  general rules about gates, signals, and connecting terminals

  E.g. $\forall t_1, t_2 \quad Connected(t_1, t_2) \Rightarrow$
  $Signal(t_1) = Signal(t_2)$

  with atomic facts given the actual circuit

  E.g., $Type(A2) = AND$.

Allows us to **reason** about overall behavior.

E.g. What inputs give a particular output.

Used in analysis of circuits/systems.

Contrast with Truth table method.

And querying (What logically follows
from specification.).

Diagnosis

So far, discussed checking specification.

How about diagnosing circuit?

I.e., given inputs and outputs, find faulty gates.

Why bother?

Don't you normally just throw the chip out??

A. Still need **good** testing patterns.

I.e., ones with large coverage,

Method applicable at higher level, i.e., in terms of larger components.

Also other domains:

- Cassini mission to Saturn (on its way...)
  real-time diagnosis / **25 msec decisions!**
  2,000 vars and 10,000 clauses SAT problem
- Software diagnosis / synthesis
  Components are subroutines / classes.

Many different possibilities...

In practice, rank according to likelihood.

E.g. single-fault more likely than double fault.

Also, incorportate failure rates of components.

(reasoning with uncertainty)

For example, what happens if we assert with our initial KB that:

$Signal(In(1, C1)) = On, \quad Signal(In(2, C1)) = On$

$Signal(In(3, C1)) = On$

$Signal(Out(1, C1)) = On, \quad Signal(Out(2, c1)) = Off$?

Need to capture possibly faulty behavior.

$$4') \ \forall g \ \ ((Type(g) = OR) \wedge Functioning(g)) \Rightarrow$$
$$(Signal(Out(1,g)) = On \Leftrightarrow$$
$$\exists n \ \ Signal(In(n,g) = On)$$

Slide CS472–9

KB with $\forall x \quad Functioning(x)$ with

$Signal(In(1,C1)) = On, \quad Signal(In(2,C1)) = On$
$Signal(In(3,C1)) = On$

implies

$Signal(Out(1,C1)) = On, \quad Signal(Out(2,c1)) = On.$

Slide CS472–10

So, if we **observe**
$Signal(Out(1, C1)) = On, \quad Signal(Out(2, c1)) = Off.$

It follows that one or more of the $Functioning()$ predicates must be **false**.

Finding the smallest set of non-functioning components consistent with the observations is called **abductive** or **diagnostic** reasoning.

Slide CS472–11

Which component(s) is (are) non-functioning?

Aside: abductive reasoning is generally computationally harder than deductive reasoning.

Still, currently abductive reasoning more widely applied because of the need for constant monitoring of complex systems.

Slide CS472–12

One more example: Graph Coloring

Graph: N nodes, K colors.

- 1) $\forall i \;\; (1 \leq i \leq N) \;\; \exists j \;\; (1 \leq j \leq K) \;\; Color(i, j)$
  $\forall i, j, l \;\; (1 \leq i \leq N) \;\; (1 \leq j, l \leq K)$
  $[(Color(i, j) \wedge Color(i, l)) \Rightarrow (j = l)]$
- 2) $\forall i, j \;\; (1 \leq i, j \leq N) \;\; [(i \neq j) \Rightarrow$
  $(Edge(i, j) \Rightarrow [\neg\exists \;\; k(1 \leq k \leq K)$
  $((Color(i, k) \wedge Color(j, k))])]$

Slide CS472–13

alternative:

- 3) $\forall i, j \;\; (1 \leq i, j \leq N) \;\; [(i \neq j) \Rightarrow$
  $(Edge(i, j) \Rightarrow [\forall \;\; k(1 \leq k \leq K)$
  $(\neg Color(i, k) \vee \neg Color(j, k))])]$

Now actual graph given by, e.g.,:

- 4) $Edge(1, 3), Edge(2, 4), Edge(5, 6)\ldots$ etc.

Slide CS472–14

reasoning: 3 & 4 gives e.g.:

$\forall\ k(1 \leq k \leq K)\ (\neg Color(1,k) \vee \neg Color(3,k))$

uses "unification" $\{i/1, j/3\}$with Modus Ponens (p. 269 R&N)

For K = 5, we get:

$(\neg Color(1,1) \vee \neg Color(3,1)), (\neg Color(1,2) \vee \neg Color(3,2)),$

$\ldots (\neg Color(1,5) \vee \neg Color(3,5))$

in propositional form.

uses Universal Elimination, e.g., substitute $\{k/1\}$, etc.

Slide CS472–15

Defining **natural kinds** is much more difficult.

e.g. a *game*, or a *chair*.

difficulty with necessary and sufficient conditions.

p. 232 R&N.

problem with "strict definition" (Quine 1953)

"the Pope is a bachelor."

Approaches?

Slide CS472–16

Inference

We've considered various first-order formalizations.

But, how do we reason with them? Derive new info?

A. Use **resolution** as in propositional case

From $(\alpha \vee p) \wedge (\neg p \vee \beta)$
conclude $\alpha \vee \beta$ until you reach contradiction.

Need some extra "tricks" to deal with
quantifiers and **variables.**

Resolution

I **put in clausal form**
all variables universally quantified
main trick: "Skolemization" to remove existantials.
idea: invent names for unkown objects known to exist

II **use unification** to match atomic sentences

III **apply resolution rule** to the clausal set combined
with negated goal. Attempt to generate empty clause.

Tricks

- **unification**: needed to match variables and terms
  between clauses that look similar
  See R&N pp. 270-271.
- **normalization**: put in **clausal form**
  move quantifiers / $\wedge$ / $\vee$ etc.
  and **Skolemization** — remove $\exists$ by giving
  an arbitrary, but unique name to the object in question.
    E.g. D for the dog owned by Jack.
    See R&N pp. 281-282.

Slide CS472–19

Unification

UNIFY (P,Q) takes two atomic sentences P and Q and returns a substitution that makes P and Q **look the same.**

Rules for substitutions:

- Can replace a variable by a constant.
- Can replace a variable by a variable.
- Can replace a variable by a function expression, as long as the function expression does not contain the variable.

Unifier: a substitution that makes two clauses resolvable.
$v_1 \rightarrow C; v_2 \rightarrow v_3; v_4 \rightarrow f(...)$

Slide CS472–20

Unification — Purpose

Given:

$Knows(John, x) \rightarrow Hates(John, x)$
$Knows(John, Jim)$

Derive

$Hates(John, Jim)$

Need **unifier** $\{x/Jim\}$ before resolution.
(simplest case)

Slide CS472–21

$\neg Knows(John, x) \vee Hates(John, x)$ and $Knows(John, Jim)$

How do we resolve? First, match them.
Solution:

UNIFY($Knows(John, x)$,$Knows(John, Jim)$) $= \{x/Jim\}$

Gives

$\neg Knows(John, Jim) \vee Hates(John, Jim)$ and
$Knows(John, Jim)$

Conclude by resolution

$Hates(John, Jim)$

Slide CS472–22

Unification (example)

general rule:

$Knows(John, x) \rightarrow Hates(John, x)$

facts:

$Knows(John, Jim)$
$Knows(y, Leo)$
$Knows(y, Mother(y))$
$Knows(x, Jane)$

"matching facts to general rules"

Slide CS472–23

UNIFY($Knows(John, x)$,$Knows(John, Jim)$) $= \{x/Jim\}$

UNIFY($Knows(John, x)$,$Knows(y, Leo)$) $= \{x/Leo, y/John\}$

UNIFY($Knows(John, x)$,$Knows(y, Mother(y))$) $=$
$\{y/John, x/Mother(John)\}$;

UNIFY($Knows(John, x)$,$Knows(x, Jane)$) $= fail$

Slide CS472–24

- Last one fails because x can't take on both the value John **and** the value Jane. But intuitively we know that everyone John knows he hates and everyone knows Jane so we should be able to infer that John hates Jane.
- This is why we require, if possible, that every variable has a separate name. Knows(John,x) and Knows(y,Jane) works.

Slide CS472–25

Most General Unifier

In cases where there is more than one substitution choose the one that makes the least commitment (most general) about the bindings.

UNIFY($Knows(John, x), Knows(y, z)$)

$= \{y/John, x/z\}$

or $\{y/John, x/z, z/Freda\}$

or $\{y/John, x/John, z/John\}$

or ....

See page 303 for general unification algorithm. $O(n^2)$

Slide CS472–26

Normal form: Clausal

Slide CS472–27

See pages 281 & 282, R&N.

- **Eliminate implication**

  $p \Rightarrow q$ becomes $\neg p \vee q$
- **Move $\neg$ inwards**

  e.g., $\neg(p \vee q)$ becomes $(\neg p \wedge \neg q)$

  $\neg \exists x \,.\, p$ becomes $\forall x \; \neg p$

  $\neg \forall x \,.\, p$ becomes $\ldots$
- **Standarize variables**

  rename variables to avoid conflicts.
- **Move quantifiers left**

  e.g., $p \vee \forall x \; q$ becomes $\forall x \; (p \vee q)$

Slide CS472–28

- **Skolemize** (remove existentials)
  e.g. $\forall x\ Person(x) \Rightarrow \exists y\ Heart(y) \wedge Has(x,y)$
  consider:
  $\forall x\ Person(x) \Rightarrow Heart(H) \wedge Has(x,H)$
  problem??
  $\forall x\ Person(x) \Rightarrow Heart(F(x)) \wedge Has(x,F(x))$
- **Distribute** $\wedge$ **over** $\vee$
  $(a \wedge b) \vee c$ becomes $(a \vee c) \wedge (b \vee c)$
- **Flatten nested conjunctions and disjunctions**
  e.g. $(a \vee b) \vee c$ becomes $(a \vee b \vee c)$

Example

Jack owns a dog.

Every dog owner is an animal lover.

No animal lover kills an animal.

Either Jack or Curiosity killed the cat, who is named Tuna.

Did Curiosity kill the cat?

Original Sentences (Plus Background Knowledge)

1. $\exists x : Dog(x) \wedge Owns(Jack, x)$
2. $\forall x \ \ (\exists y \ \ Dog(y) \wedge Owns(x, y)) \rightarrow AnimalLover(x)$
3. $\forall x \ \ AnimalLover(x) \rightarrow \forall y \ \ Animal(y) \rightarrow \neg Kills(x, y)$
4. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$
5. $Cat(Tuna)$
6. $\forall x \ \ Cat(x) \rightarrow Animal(x)$

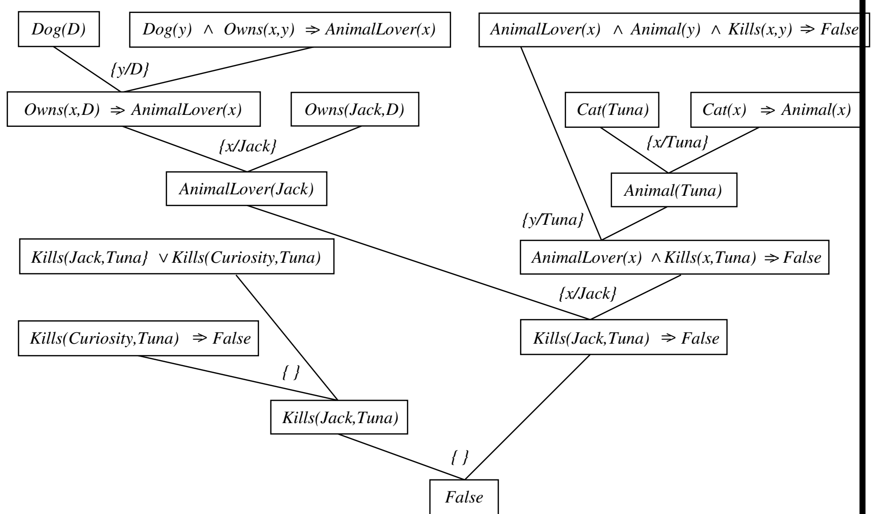
Slide CS472–31

Clausal Form

1. $Dog(D)$ (D is the function that finds Jack's dog)
2. $Owns(Jack, D)$
3. $\neg Dog(S(x)) \vee \neg Owns(x, S(x)) \vee AnimalLover(x)$
4. $\neg AnimalLover(w) \vee \neg Animal(y) \vee \neg Kills(w, y)$
5. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$
6. $Cat(Tuna)$
7. $\neg Cat(z) \vee Animal(z)$

Slide CS472–32

Proof by Resolution with Refutation



Completeness

Resolution with **unification** applied to **clausal form**, is **refutation** complete.

Interesting proof! Based on building an "artificial" domain of interpretation, called the **Herbrand universe**. See Section 9.7 R&N.

Practice

Complete in principle, in practice still significant
search problem!

Many different search strategies: **resolution strategies**