16.

The Tradeoff between Expressiveness and Tractability

Limit expressive power?

Defaults, probabilities, *etc.* can all be thought of as extensions to FOL, with obvious applications

Why not strive for the *union* of all such extensions? all of English?

Problem: automated reasoning

Lesson here:

> reasoning procedures required for more expressive languages may not work very well in practice

Tradeoff: expressiveness vs. tractability

Overview:
- a Description Logic example
- limited languages
- the problem with cases
- vivid reasoning as an extreme case
- less vivid reasoning
- hybrid reasoning systems

Simple description logic

Consider the language FL defined by:

<concept> ::= atom
  | [AND <concept> ... <concept>]
  | [ALL <role> <concept>]
  | [SOME <role>] (= [EXISTS 1 <role>])

<role> ::= atom
  | [RESTR <role> <concept>]

Example: [ALL :Child [AND Female Student]]

an individual whose children are female students

[ALL [RESTR :Child Female] Student]

an individual whose female children are students

there may or may not be male children and they may or may not be students

Interpretation $\mathfrak{I} = \langle D, I \rangle$ as before, but with

I[[RESTR r c]] = $\{ (x,y) \mid (x,y) \in I[r]$ and $y \in I[c] \}$

So [RESTR :Child Female] is the :Child relation restricted to females = :Daughter

Subsumption defined as usual

Computing subsumption

First for FL⁻= FL without the RESTR operator

- put the concepts into normalized form
- to see if C subsumes D make sure that
  1. for every $p \in C$, $p \in D$
  2. for every [SOME r] $\in C$, [SOME r] $\in D$
  3. for every [ALL s c] $\in C$, find an [ALL s d] $\in D$ such that c subsumes d.

[AND p_1 ... p_k
 [SOME r_1] … [SOME r_m]
 [ALL s_1 c_1] … [ALL s_n c_n]]

Can prove that this method is sound and complete relative to definition based on interpretations

Running time:

- normalization is $O(n^2)$
- structural matching: for each part of C, find a part of D. Again $O(n^2)$

What about all of FL, including RESTR?

Subsumption in FL

- cannot settle for part-by-part matching

  [ALL [RESTR :Friend [AND Male Doctor]] [AND Tall Rich]]

  subsumes

  [AND [ALL [RESTR :Friend Male] [AND Tall Bachelor]]
  [ALL [RESTR :Friend Doctor] [AND Rich Surgeon]]]

- complex interactions

  [SOME [RESTR *r* [AND *a b*]]]

  subsumes

  [AND [SOME [RESTR *r* [AND *c d*]]] [ALL [RESTR *r c*] [AND *a e*]]
  [ALL [RESTR *r* [AND *d e*]] *b*]]

In general: FL is powerful enough to encode *all* of propositional logic.

There is a mapping Ω from CNF wffs to FL where

$\models (\alpha \supset \beta)$ iff $\Omega(\alpha)$ is subsumed by $\Omega(\beta)$

But $\models (\alpha \supset (p \wedge \neg p))$ iff α is unsatisfiable

Conclusion: there is no good algorithm for FL unless P=NP

Moral

Even small doses of expressive power can come at a significant computational price

Questions:

- what properties of a representation language control its difficulty?
- how far can expressiveness be pushed without losing good algorithms
- when is easy reasoning adequate for KR purposes?

These questions remain unanswered, but some progress:

- need for case analyses is a major factor
- tradeoff for DL languages is reasonably well understood
- best addressed (perhaps) by looking at working systems

Useful approach:

- find reasoning tasks that are tractable
- analyze difficulty in extending them

Limited languages

Many reasoning problems that can be formulated in terms of FOL entailment (KB $\models^?$ α) admit very specialized methods because of the restricted form of either KB or α

although problem could be solved using full resolution, there is no need

Example 1: Horn clauses

- SLD resolution provides more focussed search
- in propositional case, a linear procedure is available

Example 2: Description logics

Can do DL subsumption using Resolution

Introduce predicate symbols for concepts, and "meaning postulates" like

$$\forall x[P(x) \equiv \forall y(\text{Friend}(x,y) \supset \text{Rich}(y)) \\ \wedge \forall y(\text{Child}(x,y) \supset \\ \forall z(\text{Friend}(y,z) \supset \text{Happy}(z)))]$$

[AND [ALL :Friend Rich]
[ALL :Child
[ALL :Friend Happy]]]

Then ask if MP $\models \forall x[P(x) \supset Q(x)]$

Equations

Example 3: linear equations

Let E be the usual axioms for arithmetic:

$\forall x \forall y(x+y = y+x),\ \forall x(x+0 = x),\ \ldots$ Peano axioms

Then we get the following:

$$E \models (x+2y=4 \wedge x-y=1) \supset (x=2 \wedge y=1)$$

Can "solve" linear equations using Resolution!

But there is a much better way:

Gauss-Jordan method with back substitution

- – subtract (2) from (1): $3y = 3$
- – divide by 3: $y = 1$
- – substitute in (1): $x = 2$

In general, a set of linear equations can be solved in $O(n^3)$ operations

This idea obviously generalizes!

always advantageous to use a specialized procedure when it is available, rather than a general method like Resolution

When is reasoning hard?

Suppose that instead of linear equations, we have something like

$(x+2y=4 \;\vee\; 3x-y=7) \;\wedge\; x-y=1$

Can still show using Resolution: $y > 0$

To use GJ method, we need to split cases:

$x+2y=4 \wedge x-y=1$ ß $y=1$

$3x-y=7 \wedge x-y=1$ ß $y=2$

$\therefore\; y > 0$

What if 2 disjunctions? $(eqnA_1 \vee eqnB_1) \;\wedge\; (eqnA_2 \vee eqnB_2)$

there are *four* cases to consider with GJ method

What if n binary disjunctions? $(eqnA_1 \vee eqnB_1) \;\wedge \ldots \wedge\; (eqnA_n \vee eqnB_n)$

there are 2^n cases to consider with GJ method

with $n=30$, would need to solve 10^9 systems of equations!

Conclusion: case analysis is still a *big* problem.

Question: can we avoid case analyses??

Expressiveness of FOL

Ability to represent <u>incomplete</u> knowledge

$P(a) \vee P(b)$	but which?
$\exists x\, P(x)$	$P(a) \vee P(b) \vee P(c) \vee \ldots$
and even	
$c \neq 3$	$c=1 \vee c=2 \vee c=4 \vee \ldots$

Reasoning with facts like these requires somehow “covering” all the implicit cases

languages that admit efficient reasoning do *not* allow this type of knowledge to be represented

- Horn clauses,
- description logics,
- linear equations, ...

only limited forms of disjunction, quantification etc.

Complete knowledge

One way to ensure tractability:

somehow restrict contents of KB so that reasoning by cases is not required

But is <u>complete</u> knowledge enough for tractability?

suppose KB |= α or KB |= $\neg\alpha$, as in the CWA

Get: queries reduce to KB |= ρ, literals

But: it can still be hard to answer for literals

Example: KB = $\{(p \vee q), (\neg p \vee q), (\neg p \vee \neg q)\}$

Have: KB |= $\neg p \wedge q$ complete!

But to find literals may require case analysis

So complete knowledge is not enough to avoid case analyses if the knowledge is “hidden” in the KB.

Need a form of complete knowledge that is more explicit...

Vivid knowledge

Note: If KB is complete and consistent, then it is satisfied by a *unique* interpretation I

Why? define I by I |= p iff KB |= p

ignoring quantifiers for now

Then for any I^*, if I^* |= KB then I^* agrees with I on all atoms p

Get: KB |= α iff I |= α

entailments of KB are sentences that are true at *I*

explains why queries reduce to atomic case

$(\alpha \vee \beta)$ is true iff α is true or β is true, *etc.*

if we have the I, we can easily determine what is or is not entailed

Problem: KB can be complete and consistent, but unique interpretation may be hard to find

Solution: a KB is <u>vivid</u> if it is a complete and consistent set of literals (for some language)

e.g. KB = $\{\neg p, q\}$ specifies I directly

Quantifiers

As with the CWA, we can generalize the notion of vivid to accommodate queries with quantifiers

A first-order KB is vivid iff for some finite set of positive function-free ground literals KB$^+$, KB = KB$^+$ $\cup$ *Negs* $\cup$ *Dc* $\cup$ *Un.*

Get a simple recursive algorithm for KB $\models \alpha$:

KB $\models \exists x.\alpha$ iff KB $\models \alpha[x/c]$, for some $c \in$ KB$^+$

KB $\models (\alpha \vee \beta)$ iff KB $\models \alpha$ or KB $\models \beta$

KB $\models \neg\alpha$ iff KB $\not\models \alpha$

KB $\models (c = d)$ iff c and d are the same constant

KB $\models p$ iff $p \in$ KB$^+$

This is just database retrieval

- useful to store KB$^+$ as a collection of relations
- only KB$^+$ is needed to answer queries, but *Negs, Dc,* and *Un* are required to *justify* the correctness of the procedure

Analogues

Can think of a vivid KB as an analogue of the world

there is a 1-1 correspondence between

- objects in the world and constants in the KB$^+$
- relationships in the world and syntactic relationships in the KB$^+$

for example, if constants c_1 and c_2 stand for objects in the world o_1 and o_2

there is a relationship R holding between objects o_1 and o_2 in the world

iff

constants c_1 and c_2 appear as a tuple in the relation represented by R

Not true in general

for example, if KB = $\{P(a)\}$ then it only uses 1 constant, but could be talking about a world where there are 5 individuals of which 4 satisfy P

Result: certain reasoning operations are easy

- how many objects satisfy P (by counting)
- changes to the world (by changes to KB$^+$)

Beyond vivid

Requirement of vividness is very strict.

Want weaker alternatives with good reasoning properties

Extension 1

ignoring quantifiers again

Suppose KB is a finite set of literals

- not necessarily a complete set (no CWA)
- assume consistent, else trivial

Cannot reduce KB |= α to literal queries

if KB = $\{p\}$ then KB |= $(p \wedge q \vee p \wedge \neg q)$ but KB $|\neq p \wedge q$ and KB $|\neq p \wedge \neg q$

But: assume α is small. Can put into CNF

α ß $(c_1 \wedge ... \wedge c_n)$

- KB |= α iff KB |= c_i, for every clause in CNF of α
- KB |= c iff c has complimentary literals – tautology
  or KB $\cap$ c is not empty

Extension 2

Imagine KB vivid as before + new definitions:

$\forall xyz[R(x,y,z) \equiv$ ... wff in vivid language ...]

Example: have vivid KB using predicate ParentOf

add: $\forall xy[\text{MotherOf}(x,y) \equiv \text{ParentOf}(x,y) \wedge \text{Female}(x)]$

To answer query containing $R(t_1,t_2,t_3)$, simply macro expand it with definition and continue

- can handle arbitrary logical operators in definition since they become part of *query*, not KB
- can generalize to handle predicates not only in vivid KB, provided that they bottom out to KB$^+$

  $\forall xy[\text{AncestorOf}(x,y) \equiv \text{ParentOf}(x,y) \vee$
  $\exists z\ \text{ParentOf}(x,z) \wedge \text{AncestorOf}(z,y)]$

- clear relation to Prolog

  a version of logic programming based on inductive definitions, not Horn clauses

Other extensions

Vivification: given non-vivid KB, attempt to make vivid e.g. by eliminating disjunctions *etc.*

for example,

- use taxonomies to choose between disjuncts
  - Flipper is a whale or a dolphin.
- use intervals to encompass disjuncts
  - The picnic will be on June 2, 3,or 4th.
- use defaults to choose between disjuncts
  - Serge works in Toronto or Montreal.

Problem: what to do with function symbols, when Herbrand universe is not finite?

partial Herbrand base?

Hybrid reasoning

Want to be able to incorporate a number of special-purpose efficient reasoners into a single scheme such as Resolution

Resolution will be the glue that holds the reasoners together

Simple form: <u>semantic attachment</u>

- attach procedures to functions and predicates
  - e.g. numbers: procedures on plus, LessThan, ...
- ground terms and atomic sentences can be *evaluated* prior to Resolution
  - *P(*factorial*(4),* times*(2,3))* ß *P(24, 6)*
  - LessThan*(*quotient*(36,6), 5)* ∨ α ß α
- much better than reasoning directly with axioms

More complex form: <u>theory resolution</u>

- build theory into unification process (the way paramodulation builds in =)
- extended notion of complimentary literals
  - $\{\alpha,$ LessThan$(2,x)\}$ and $\{$LessThan$(x,1), \beta\}$ resolve to $\{\alpha,\beta\}$

Using descriptions

Imagine that predicates are defined elsewhere as concepts in a description logic

Married $\doteq$ [AND ...]　　　　Bachelor $\doteq$ [AT-MOST ...]

then $\{P(x), \text{Married}(x)\}$ and $\{\text{Bachelor(john)}, Q(y)\}$ resolve to $\{P(\text{john}), Q(y)\}$

Can use description logic procedure to decide if two predicates are complimentary

instead of explicit meaning postulates

Residues: for “almost” complimentary literals

$\{P(x), \text{Male}(x)\}$ and $\{\neg\text{Bachelor(john)}, Q(y)\}$

resolve to

$\{P(\text{john}), Q(y), \text{Married(john)}\}$

since the two literals are contradictory *unless* John is married

Main issue: what resolvents are necessary to get the same conclusions as from meaning postulates?

residues are necessary for completeness