- Project #2 - should be underway....

> If the validity of a small part of the program depends on properties that can only be established by a scan of the program as a whole, then you know you've done a bad job as a language designer, and you don't need your customers to tell you that. ... In fact customers don't tell you – it's very easy to persuade your customers that anything that goes wrong is their fault and not yours.

Tony Hoare, Null References: The Billion Dollar Mistake, 2009
`https://www.infoq.com/presentations/Null-References-The-Billion-Dollar-Mistake-Tony-Hoare/`

Plan

Since Midterm

- difference lists, definite clause grammars and natural language interfaces to databases
- computer algebra and calculus
- Triples are universal representations of relations, and are the basis for RDF, and knowledge graphs
- URIs/IRIs provide constants that have standard meanings
- Ontologies define the meaning of symbols used in information systems.
- You should know what the following mean: RDF, IRI, rdf:type, rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain, rdfs:range

Today

- ontologies in science
- Complete Knowledge Assumption and Negation as failure

Knowledge Sharing

- One ontology typically imports and builds on other ontologies.
- OWL provides facilities for version control.
- Tools for mapping one ontology to another allow inter-operation of different knowledge bases.
- The semantic web promises to allow two pieces of information to be combined if
  - they both adhere to an ontology
  - these are the same ontology or there is a mapping between them.

Curing Cancer: Data in Science

- PubMed (`https://pubmed.ncbi.nlm.nih.gov`) contains 36 million citations for biomedical literature. About 2500 citations are added each day (curated with funding, genetic, chemical and other metadata.)
- Papers typically present some data, and give a description of the results.
- Many paper provide a meta-analysis, which collects results from multiple related studies and explain what can be learned from them.
- Some have suggested using chat-GPT to read the text and answer questions about them them. (Note that each author has very limited knowledge).
- An alternative is to combine the data to provide aggregated results. Challenge: interoperation of datasets.

Meta-data: Provenance

- The provenance of data or data lineage specifies where the data came from and how it was manipulated
- Provenance is typically recorded as metadata – data about the data – including:
  - Who collected each piece of data? What are their credentials?
  - Who transcribed the information?
  - What was the protocol used to collect the data? Was the data chosen at random or chosen because it was interesting or some other reason?
  - What were the controls? What was manipulated, when?
  - What sensors were used? What is their reliability and operating range?
  - What processing has been done to the data?

FAIR data

FAIR principles for data:

- *Findable* – the (meta)data uses unique persistent identifiers, such as IRIs.
- *Accessible* – the data is available using free and open protocols, and the metadata is accessible even when the data is not.
- *Interoperable* – the vocabulary is defined using formal knowledge representation languages (ontologies).
- *Reusable* – the data uses rich metadata, including provenance, and an appropriate open license, so that the community can use the data.

Ontologies in Science

- https://schema.org
- SNOMED-CT is a medical ontology of clinical terms that defines over 350,000 concepts in multiple languages. https://www.snomed.org/snomed-ct/five-step-briefing
- http://obofoundry.org
- https://www.springernature.com/gp/open-research/open-data

Prolog Limitations

- Suppose you had a database using the relation:

  $$enrolled(S, C)$$

  which is true when student S is enrolled in course C.
- Can you define the relation:

  $$empty\_course(C)$$

  which is true when course C has no students enrolled in it?
- Why? or Why not?
  $empty\_course(C)$ doesn't logically follow from a set of *enrolled* relation because there are always models where someone is enrolled in a course!

Complete Knowledge Assumption (CKA)

- Often you want to assume that your knowledge is complete. Everything not known to be true is false.
- Example: you can state what switches are up and the system can infer that the other switches are down.
- Example: assume that a database of TA hours is complete.
- The definite clause language is monotonic: adding clauses can't invalidate a previous conclusion.
- Under the complete knowledge assumption, the system is non-monotonic: adding clauses can invalidate a previous conclusion.
- The complete knowledge assumption is sometimes called the closed world assumption.

Completion of a knowledge base

- Suppose the rules for atom a are

$$a \;{:}\!-\; b_1.$$
$$\vdots$$
$$a \;{:}\!-\; b_n.$$

  equivalent logical formula $a \;{:}\!-\; b_1 \vee \ldots \vee b_n$.
  "a is true if b_1 or ... or b_n"
- Under the Complete Knowledge Assumption, if a is true, one of the b_i must be true:

$$a \rightarrow b_1 \vee \ldots \vee b_n.$$

  "a implies b_1 or ... or b_n"
- Under the CKA, the clauses for a mean Clark's completion:

$$a \leftrightarrow b_1 \vee \ldots \vee b_n$$

  "a is true if and only if b_1 or ... or b_n"

Clark's Completion of a KB

- Clark's completion of a knowledge base consists of the completion of every atom.
- An atom h with no clauses, has the completion
  $h \leftrightarrow false$.
  "h is false".
- You can interpret negations in the body of clauses.

  $$\backslash+ \ h$$

  means that h is false under the complete knowledge assumption
  This is negation as failure.
  $\vdash$ means can be proved
  $\nvdash$ means cannot be proved
  \+ looks like it

Electrical Environment (`elect_naf.pl`)

Idea: only represent `up` and use `\+ up` instead of `down`

- Easier to specify
- Less error prone (exactly one must be true)

Negation as failure example (naf.pl)

```
p :- q, \+ r.
p :- s.
q :- \+ s.
r :- \+ t.
t.
s :- w.
```

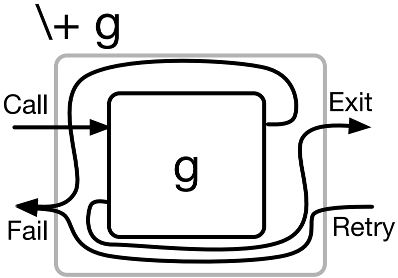
Bottom-up negation as failure interpreter

$C := \{\}$
repeat
    either
        select $r \in KB$ such that
            r is "h :- $b_1, \ldots, b_m$"
            $b_i \in C$ for all i, and
            $h \notin C$
        $C := C \cup \{h\}$
    or
        select h such that for every rule "h :- $b_1, \ldots, b_m$" $\in KB$
                either for some b_i, \+ $b_i \in C$
                or some $b_i =$ \+ g and $g \in C$
        $C := C \cup \{$ \+ $h\}$
until no more selections are possible

Negation as failure example (naf.pl)

```
p :- q, \+ r.
p :- s.
q :- \+ s.
r :- \+ t.
t.
s :- w.
```

Box Model of Negation-as-failure



Top-Down negation as failure proof procedure

- If the proof for h fails, you can conclude $\backslash+\ h$.
- Failure can be defined recursively:
  Suppose you have rules for atom h:

  $$\begin{array}{l} h \ \text{:-}\ b_1 \\ \quad\vdots \\ h \ \text{:-}\ b_n \end{array}$$

  If every body b_i fails, h fails.
  A body fails if one of the conjuncts in the body fails.
- Special case: if there are no rules for h then h fails

Example: default reasoning about resorts (beach.pl)

- A resort is on the beach or away from the beach.
  A resort is away from the beach unless it says it is on a beach.

```
away_from_beach :- \+ on_beach.
```

- If we are told the resort is on the beach, we would expect that resort users would have access to the beach.
  If they have access to a beach, we would expect them to be able to swim at the beach.

```
beach_access :- on_beach, \+ ab_beach_access.
swim_at_beach :- beach_access, \+ ab_swim_at_beach.

ab_swim_at_beach :- enclosed_bay, big_city, \+ ab_no_swim
ab_no_swim :- in_BC, \+ ab_BC_beaches.
```

http://cs.ubc.ca/~poole/cs312/2024/prolog/beach.pl

Negation as Failure and the Unique Names Assumption

- Suppose a knowledge base contains the single fact about p:

  ```
  p(a).
  ```

- What happens to the query `\+ p(b)`?
- What does this mean about the relationship between a and b?
- With negation as failure, Prolog assumes the unique names assumption: different ground (variable-free) terms denote different individuals.
- In "Pure Prolog" without negation-as-failure (and related concepts, e.g., counting) a query logically follows if it follows whether names are unique or not.

Unique Names Assumption

- Suppose the only clauses for *enrolled* are

  $enrolled(sam, cs222)$

  $enrolled(chris, cs222)$

  $enrolled(sam, cs873)$

  To conclude $\neg enrolled(chris, cs873)$, what do we need to assume?
  - All other enrolled facts are false
  - Inequalities:

  $$sam \neq chris \wedge cs873 \neq cs222$$

- The unique names assumption (UNA) is the assumption that distinct ground terms denote different individuals.