More language

Sapir-Whorf, eat your heart out
Language design basics
The Design Triangle (recalled)

Expressivity
(Representational Adequacy)

Usability
(Weak Cognitive Adequacy vs. Cognitive Complexity)

Computability
(vs. Computational and Implementational Complexity)

Each vertex has multiple positive and negative interactions with the others
The Design Triangle (recalled)

- **Expressivity**
  - What we can *say* (or say easily, or naturally, or...)
  - If suitable to our needs, a formalism (or KR) is representationally adequate
- **Computational Complexity**
  - How *hard* is it (in terms of resources) to work with?
  - Related: Implementational Complexity
    - How hard is it to produce a *production quality* implementation
- **Cognitive Complexity**
  - Focus on Weak Cognitive Adequacy i.e., *Usability*
- A good KR (or KR formalism) achieves a *good balance* of all of these for *most of its uses, most of the time*
Language Design

- KR languages are like (individual) KRs
  - Indeed, we can view them as a (v. general) KR
  - The design considerations are similar

- General desiderata:
  1. Clarity of specification
  2. Expressivity
  3. Usability
  4. Computability

- As we’ve seen, 1. points toward logic!
  - FOL (at least) is well understood
  - FOL (at least) is v. expressive
    * Though it has known holes
  - Usability....
  - Computability...

The usual triad
Some Considerations

• Designing a logic from a reasoning perspective
  – What expressivity do you need?
  – What's your core service?
  – What are the key services?
  – Are you interactive or not?
  – What's the scale you need to deal with?
    • And other performance characteristics
  – What do you know about implementation?

• What these somewhat neglect
  – Many surface syntax issues
  – Non logical aspects of the language
  – Cognitive complexity
Automated Reasoning (types)

• Automated Reasoning
  – (Sometimes known: “theorem proving”)  
  – (Typically: Deduction!)  
  – Two basic flavors

• Interactive reasoning:
  – “Proof assistants”
  – A human user interacts with the program to generate proofs
  – System provides verification of steps, suggested next steps, etc.
  – “Mixed initiative”

• (Fully) Automated Reasoning
  – Set up initial conditions
    • Premises, question to be answered
      – Default questions
  – Program does the rest!
Automated Reasoning (which one?)

- What's your goal?
- Interactive (typical)
  - Prove or verify math theorems
  - Verify algorithms
  - Things which are v. hard to prove
    - Because there isn’t any fully automated way
- Automated (typical)
  - Some math theorem proving
  - Whenever your concern is for the answer than the process
    - Database integration
    - Expert systems
    - Query answering
    - Debugging representations
Interactive Reasoning vs. System

• Interactive reasoning refers to the process of proving
  – Human intervention is required to supply proof steps

• AR can be part of an interactive system
  – E.g., a dialogue driven expert system
  – Consider database based systems
  – Consider Protege!
Services

• Core
  – Satisfiability/Consistency Checking

• Key
  – Entailment
    • With default questions
      – Classification (atomic subsumptions)
        » Atomic class satisfiability
    – Instantiation
      • With different ASK languages
  – More outré
    • Explanations of entailments
    • Modules

• What do the core services give us?
  – Verification, reaction, query
Default Questions

• Good default questions (can) improve usability
  – User does not need to formulate the questions
  – Easy invocation of the questions
  – Built in coverage
    • Classification checks EVERY pair of atomic classes for subsumption
      – Given \{A, B, C\},
        » check A SubClassOf: B; A SubClassOf: C, B SubClassOf:C...
    • How many checks?

• Even good default questions can cause problems
  – Formulation takes practice and inclination
    • Skills atrophy!
  – Users write to the default queries
    • May distort modelling
      • In serious ways!
  – Reasoners get overtuned to default questions
Computability

• What issue are we facing?
  – For some problem
    • Say consistency checking (CC)
  – is there a program that solves that problem?
    • and for us, in “reasonable” time/space
    • i.e., with reasonable complexity

• First big question:
  – Can we write a program s.t.
    • for any KB encoded in FOL
    • return YES if KB is consistent or NO if KB is inconsistent
    • in finite time
  – Answer: NO
  – FOL is only semi-decidable
Decidability

• A decidable problem:
  – For any (finite) string $S$ and (finite) regular expression $R$
    • answer YES if $R$ matches $S$ and NO if $R$ does not match $S$
  – $(A^*B|B^*A)$
    • “AAAAA”
    • “AAAAB”
    • “ABA”
    • “BA”
    • “”

• Why?
  – (Hint, regular expressions correspond to deterministic finite state machines.)
  – (With some work!)
  – Linear time, constant space
Complexity

• **How much time (or space)?**
  – As a function of the input
    • Asymptotic complexity.
  – Complexity of problem vs. of algorithm
    • How hard is the problem?
    • How efficient is the algorithm?
    • Consider SAT for FOL vs. PL
      – Problem
        » SAT for FOL is semi-decidable (only)
        » SAT for PL is decidable!
      – Algorithm
        » Can I use FOL to determine PL SAT?
  – Complexity of an instance vs. of a class
    • Are all instances of FOL SAT undecidable?
      – Does this make sense?
    • Are all (infinite) subsets of FOL SAT semi-decidable?
### Complexity (function)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N</td>
<td>2N</td>
<td>N^2</td>
<td>2^N</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>10</td>
<td>25</td>
<td>32</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>12</td>
<td>36</td>
<td>64</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>14</td>
<td>49</td>
<td>128</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>16</td>
<td>64</td>
<td>256</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>18</td>
<td>81</td>
<td>512</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>20</td>
<td>100</td>
<td>1024</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>22</td>
<td>121</td>
<td>2048</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>24</td>
<td>144</td>
<td>4096</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>26</td>
<td>169</td>
<td>8192</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>28</td>
<td>196</td>
<td>16384</td>
</tr>
<tr>
<td>16</td>
<td>15</td>
<td>30</td>
<td>225</td>
<td>32768</td>
</tr>
<tr>
<td>17</td>
<td>16</td>
<td>32</td>
<td>256</td>
<td>65536</td>
</tr>
</tbody>
</table>
Complexity (function

Complexity

70000
65000
60000
55000
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

2N
N^2
2^N
Complexity (function)

![Graph showing complexity functions](image)

- $2N$
- $N^2$
- $2^N$
Complexity of a problem class

• Many “NO”s
  – Within a problem class, instances vary in difficulty
  – Between two problem classes...
    • Things get tricky!

• Worst case complexity
  – WCC is the most common measure
    • Given a problem class P
      – what’s the complexity of the hardest problems in P
    • Given an algorithm A for answering P
      – what’s the worst behavior we see for A

• Does WCC(P) = WCC(A)?
• Must WWC(P) < WWC(A)?
  – Why use such an A?
Complexity of a problem class

• WCC is often uninformative
  – How often do you hit the worst case?
    • “on average”
    • “in practice”
  – What’s the best case?
• Sometimes high WCC can be beneficial
  – From a complexity POV!?
  – Tends to correspond with higher expressivity
    • Thus, terse encodings
    • Low complexity isn’t a win if you need to start with huge input
Some Logics
Names of Logics

• Description logics
  – A family of (generally) decidable (generally fragments of first order) logic[s]
  – TBox, ABox, (RBox, DataBox)
  – Different logics have
    • Different expressivity
    • Different cognitive complexity
    • Trade offs!

• TBox (historically) was the focus
  – So, DLs were characterized by their class expression language
    • or class constructors
A Base Logic

- Classically, we start with “\( \mathcal{AL} \)”
  - “Attribute logic”
  - This is the “concept” (class expression) language

<table>
<thead>
<tr>
<th>Syntax</th>
<th>DL name</th>
<th>OWL name</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>Atomic concept</td>
<td>Class name/entity</td>
</tr>
<tr>
<td>( \top )</td>
<td>Universal/top concept</td>
<td>owl:Thing</td>
</tr>
<tr>
<td>( \bot )</td>
<td>Bottom concept</td>
<td>owl:Nothing</td>
</tr>
<tr>
<td>( \neg A )</td>
<td>Atomic negation</td>
<td>complementOf</td>
</tr>
<tr>
<td>( \cap )</td>
<td>intersection/conjunction</td>
<td>intersection</td>
</tr>
</tbody>
</table>
All about \( \mathcal{AL} \)

- **Historical**
  - Logical reading of “frames”/semantic nets
    - Universal interpretation of “slots”
    - “Typing” reading
  - “Smallest” “sensible” DL
    - \( \mathcal{FL} \) is smaller :)

- **Computational**
  - Subsumption between concepts is polynomial, but...
  - ...gets much harder with (non-empty) TBoxes!

- **Orthogonality**
  - ...

- **Usability**
  - Low (universal is not the right choice, generally)
AL on the Design Triangle?

- Expressivity (Representational Adequacy)
- Usability (Weak Cognitive Adequacy vs. Cognitive Complexity)
- Computability (vs. Computational and Implementational Complexity)
A More Expressive Logic

- $\mathcal{ALC}$
  - “Attribute logic with complement”; $C$ and $D$ are expressions
  - Contains propositional logic
  - What about $\mathcal{AL} + C$?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>DL name</th>
<th>OWL name</th>
<th>“Letter”</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Atomic concept</td>
<td>Class name/entity</td>
<td></td>
</tr>
<tr>
<td>$C \cap D$</td>
<td>Intersection/conjunction</td>
<td>intersectionOf</td>
<td></td>
</tr>
<tr>
<td>$C \cup D$</td>
<td>Union/disjunction</td>
<td>unionOf</td>
<td>$U (ALU)$</td>
</tr>
<tr>
<td>$\neg C$</td>
<td>Concept negation</td>
<td>complementOf</td>
<td>$C (ALC)$</td>
</tr>
<tr>
<td>$\exists P.C$</td>
<td>Existential Restriction</td>
<td>someValuesFrom</td>
<td>$E (ALE)$</td>
</tr>
<tr>
<td>$\forall P. C$</td>
<td>Universal Restriction</td>
<td>allValuesFrom</td>
<td></td>
</tr>
</tbody>
</table>
A More Expressive Logic

- **$ALC$**
  - Concept negation brings everything else
  - $AL + C = ALC = ALU\overline{EC}$

<table>
<thead>
<tr>
<th>Syntax</th>
<th>AL + C translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$A$</td>
</tr>
<tr>
<td>$C \cap D$</td>
<td>$C \cap D$</td>
</tr>
<tr>
<td>$\forall P. C$</td>
<td>$\forall P. C$</td>
</tr>
<tr>
<td>$\neg C$</td>
<td>$\neg C$</td>
</tr>
<tr>
<td>$C \cup D$</td>
<td>$\neg (\neg C \cap \neg D)$</td>
</tr>
<tr>
<td>$\exists P. C$</td>
<td>$\neg \forall P. \neg C$</td>
</tr>
<tr>
<td>$\top$</td>
<td>$A \cup \neg A$</td>
</tr>
<tr>
<td>$\bot$</td>
<td>$A \cap \neg A$</td>
</tr>
</tbody>
</table>

$AL$ (when we add atomic negation and top)

$C$ ($ALC$)

$U$ ($ALU$)

$E$ ($ALE$)

For some new “$A$”
All about $\mathcal{ALC}$

• Smallest propositionally closed DL
  – “Boolean” DL
  – First “very expressive” DL

• Computational
  – Contains Propositional Logic so NP-Hard!
    • PSpace-Complete for Concept Satisfiability
  – TBoxes can make it EXPTIME-Complete

• Orthogonality (we saw)

• Usability
  – Not terrible
  – Still missing a ton
    • Can’t count
    • No transitivity
      – Property language is weak overall!
**ALC TBoxes**

- Two major kinds
  - “Definitorial”
    - Every TBox axiom is an equivalence
    - Every TBox axiom has at least one atomic side
    - No cycles
  - “General”
    - Any expression on either side
    - No other restrictions!

- Big jump in expressivity and complexity
  - ALC + Definitorial TBoxes is PSPACE-Complete
    - No harder than concept satisfiability!
  - ALC + General TBoxes is EXPTIME-Complete

- Axiom shape matters a lot!
  - But “definitorial” is “neat”
Two Expressivities/Complexities

• Constructors (concept expression language)
  – $AL$ vs. $ALC$
  – Not all new constructors = new expressive power
    • $ALUC$ vs. $ALEC$ vs. $ALEUC$

• Axioms and axiom “shape”
  – Non-empty TBoxes
  – Definitorial
    • Breaks the “everywhere there was a name, replace with an expression”
    • Irregular (but regularly so)
  – General
    • More uniform, but computationally harder

• Interactions betwixt the two
A Simple Example
A Case of Disjointness

• In $\mathcal{ALC}$ we can force two classes to be disjoint
  – Tree SubClassOf: not Human
  – Contrast: Tree EquivalentTo: not Human

• Slight syntactic extension: DisjointWith:
  – Tree DisjointWith: Human
  – What’s the effect on expression, computation, and cognition?
  – Issue! Common to have sets of disjoint classes
    • E.g., siblings (for covering)
    • Require $\approx n^2$ disjointness axioms for $n$ classes
    • Files dominated by disjointness axioms
      – Hard to edit
      – Hard to read
      – Significant load time issues
Pairwise Disjointness

- Sponge, Unicorn, Slug, Tree
- **Somewhat compact**
  - Sponge DisjointWith: Unicorn, Slug, Tree
  - Unicorn DisjointWith: Sponge, Slug, Tree
  - Slug DisjointWith: Sponge, Unicorn, Tree
  - Tree DisjointWith: Sponge, Slug, Unicorn,
- **More compact (exploiting semantics):**
  - Sponge DisjointWith: Unicorn, Slug, Tree
  - Unicorn DisjointWith: Slug, Tree
  - Slug DisjointWith: Tree
- **Tradeoffs for expression/computation/cognition?**
  - Do they differ in what they say?
  - In WWC? BCC? ACC?
  - Is one more usable?
N-ary Disjointness

- Introduce an n-ary construct: DisjointClasses:

  - Very compact
    - DisjointClasses: Sponge, Unicorn, Slug, Tree
    - Expression of size n for n classes
      • Must take care in measuring size!

- Rather “DRY”
  - Where does it get more complicated?
  - Does it ever get more complicated than the alternatives?

- Tradeoffs for expression/computation/cognition?
  - Does this change expressivity?
  - Change WWC? BCC? ACC?
    • What if we implement it by preprocessing into pairwise disjointness?
    • What does it do to the input?
  - Is one more usable?
Lessons Learned

• n-ary and pairwise disjointness
  – Are polynomially interreducible
    • Thus no change in the asymptotic complexity classes
    • Can have large effect in practice
  – (Potentially) Affect different parts of processing

• Big effect on cognition
  – But not 100% obvious
  – Size issues dominate
    • But, also, repetition
  – Performance effects can be high (on cognitive issues)
    • Waiting to download/load == wasted time for little gain
  – Workarounds helpful
    • But built in support best
A more complex example
Examples thus far

• Propositional vs. FOL
  – Inexpressive vs. very expressive
  – Decidable vs. semi-decidable

• vs. ALC
  – Expressivity: PL < ALC << FOL
    • (The more expressive contains the less)

• ALC with and without
  – binary disjoint axioms
  – n-ary disjointness axioms

• Different extensions and restrictions
  – Additional constructors (e.g., min/max)
  – Additional axiom types (SubPropertyOf)
  – Patterns of use or axiom shape (“definitorial” axioms)
  – Semantic restrictions (named individuals)
  – Built-in functionality (datatypes)
Two new constructors: min & max

• Consider:
  – loves some Person
  – loves min 1 Person
  – loves max 1 Person
  – loves exactly 1 Person

• More elaborate:
  – loves min 3 Person
  – loves max 2 Person
  – (loves min 3 Person) and (loves max 2 Cat)
  – (loves min 3 Person) and (loves max 2 Person)

• ALCQ
  – ALC + min and max, the “counting quantifiers”
  – Expressivity ++
  – “The same” computational complexity (more implementation burden)
  – Cognitive complexity...
Complexity interlude

- What is “having the same” complexity?
  - Having exactly the same resource function?
  - Being “polynomially reducible”
    - A problem $P$ is polynomially reducible to problem $Q$ iff
      - there is a function, $f$, s.t. for every instance of $P$, $p$
      - $f(p)$ is in $Q$
      - $|f(p)|$ is (at most) “polynomially bigger” than $|p|$
        - i.e., $|p| = \text{some polynomial over } |f(p)|$

- Consider ALC with $n$-ary disjointness (“ALnC”)
  - $f = \text{for any KB in ALnC}$
    - For each DisjointClasses: axiom
    - replace with $\approx$quadratic DisjointWith: axioms
  - Thus, ALnC is polynomially reducible to ALC
    - Thus, we don’t have a fundamental change in complexity
    - Though we might have a notable change!
New Axiom Type: Transitivity

- ALC + Transitive = S
  - loves Characteristics: Transitive
  - knows Characteristics: Transitive
  - trusts Characteristics: Transitive
  - locatedIn Characteristics: Transitive
  - partOf Characteristics: Transitive

- These can be combined with quantifiers
  - knows some Person
  - knows some (knows some Person)

- We can add another axiom type
  - S + SubPropertyOf: = SH

- No worries!
Transitivity + Counting!

- ALC + Transitive + min & max
  - partOf Characteristics: Transitive
  - (partOf exactly 4 Finger) and (part of exactly 1 Thumb)
  - partOf exactly 2 Hands
  - We’d like to infer:
    - (partOf exactly 8 Fingers) and (partOf exactly 2 Thumb)

- PROBLEM
  - ALC + Transitive + Counting = Semi-decidable
What to do?

• Give up one or the other
  – But both are very useful
  – Transitivity: Part-Of or Located-In!
  – Number Restrictions: 4 fingers and 1 thumb!

• Live with undecidability
  – ALCQ and SH are already EXPTIME Complete
    • “Practically” undecidable?
  – What is the impact on implementation design?
  – What is the impact on user interaction?

• Find a middle ground
  – Can have transitive roles and counting in the same ontology
    • But not intertwined
    • Can only count non-transitive, i.e., simple roles
  – This is the choice that OWL (DL 1&2) made
    • Pattern of use restriction
The Restriction

• Only simple roles in the scope of a number restriction
  - P exactly 1 C
  - P exactly 1 C. P Characteristics: Transitive

• An ObjectProperty is simple iff it is not transitive and has no transitive subroles
  - P exactly 1 C. Q SubPropertyOf: P.
  - P exactly 1 C. Q SubPropertyOf: P

• Fairly severe restriction!
  - Bijan hasPart exactly 2 Legs.
  - Legs SubClassOf hasPart exactly 1 Foot
  - Those feet can’t be (implied to be) part of me!
    • And I can’t count them!
Cognitive Complexity

• There are workarounds
  – Two properties: hasPart and hasDirectPart
  – hasPart is transitive
  – Only count the latter
  – Manually propagate

• Burdens
  – We don’t get what we want
    • Can’t count my thumbs!
    • We do get that my thumbs are part of me
  – Workarounds distort the modelling
  – Workarounds for lacking either are generally worse
  – Not a “motivated” or regular restriction
    • Combining two legal ontologies can yield an illegal one
      – Thus, not closed under union
## Property Expressivity

- Transitivity yields the logic $S$
- Adding role hierarchies yields $SH$
- We can generalize this with role composition

<table>
<thead>
<tr>
<th></th>
<th>Property chains</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS</td>
<td>ObjectPropertyChain ( :hasFather :hasBrother )</td>
</tr>
<tr>
<td>MS</td>
<td>hasFather o hasBrother</td>
</tr>
<tr>
<td>NL</td>
<td>my father’s brother</td>
</tr>
<tr>
<td>FOL</td>
<td>hasFather($x$, $y$) $\land$ hasBrother($y$, $z$)</td>
</tr>
<tr>
<td>DL</td>
<td></td>
</tr>
</tbody>
</table>
Constructors vs. Axiom Shape

• We said that $\mathcal{ALC}$ is EXPTIME complete
  – ...for certain kinds of TBox
  – Is PSPACE..

  • The TBox should be definitorial
    – Approx: No General Concept Inclusion (GCIs) or cycles
    – (Preferably) only acyclic definitions, with one definition per name
      – GCI: $C$ and $D$ SubClassOf: $E$ or $F$
      – Cyclic: $A$ EquivalentTo: $P$ some $A$

• A clean example
  – $A$ EquivalentTo: $P$ some $B$
  – $B$ EquivalentTo: $E$ or $F$
  – $F$ EquivalentTo: $P$ some not $G$

• Now $A$ can be replaced everywhere with:
  – $P$ some (E or (P some not G))

• “Macro expansion” or unfolding

• Care must be taken to avoid exponential blowup!
General TBoxes

• If we allow GCIs or cycles, things get harder
  – Our axioms are no longer "local"
  – We can talk about every element in the domain
  – A lot of expressivity!
  – General ALC is “as complex” as SHIQ or SHOQ!

• Conversely, SH, even with axiom restrictions, is hard
  – We can encode general TBoxes as SH concepts
    • Encode each axiom as a concept and form their conjunction, M
      – C SubClassOf: D as not C or D
        » (C and D are complex and/or cyclic)
      – M EquivalentTo: (not C1 or D1) and (not C2 or D2)...
    • Create a transitive superproperty of all properties, U
    • (M and (U only M)) captures the TBox!
Pointless Restrictions

• If we add SubProperties and Transitivity to ALC
  – There is no complexity advantage to non-general TBoxes
    • Though having to encode one’s GCI as internalized concepts sucks
    • We pay the complexity price without getting (easy) expressivity
  • If we allow multiple definitions for the same term
    – We get GCI (via transitivity of equivalence)
      • A EquivalentTo: P some D.
      • A EquivalentTo: Q some E.
      • Thus P some D EquivalentTo: Q some E.
      • Unioning definitorial ontologies is dangerous
    – This bit OWL Lite hard!
      • Was supposed to be a “easy to implement, use, and learn” fragment
      • Equivalent to at least SHIF
      • Allowed multiple definitions (among other things) for easy merge
      • “Expressively Pointless” languages not well understood!
Pointless Restrictions?

• While the extra expressivity is possible
  – It’s not obvious
  – So maybe people won’t use it
    • Lack of affordances
  – Maybe the ontologies they write will be “easier”

• Consider
  – All EXPTIME complete logics are polynominally interreducible
    • That’s what it is to be EXPTIME complete
  – So they all have “the same” expressivity
    • But general ALC is a much more cramped logic than SHIQ
    • General ALC is much easier to implement (at first blush)
  – How critical is language coverage?
    • That is, having a sound and complete procedure?
Definition Oriented Development

Define define
The Simple Development Cycle

1. Conceptualize
2. Verbalize
3. Formalize

∃P.C ⊑ ∀P.D
A∩B ⊑ ∀P.D
∃P(A∪B) ⊑ ¬D

has knowledge

has Knowledge
The Simple Development Cycle

- Conceptualize
- Communicate
- Formalize

Formalize

∃P.C ⊑ ∀P.D
A ∩ B ⊑ ∀P.D
∃P(A ∪ B) ⊑ ¬D

Wednesday, 30 November 2011
The Simple Development Cycle

Conceptualize

Communicate

Formalize

Inference

∃P.C ⊑ ∀P.D
A ∩ B ⊑ ∀P.D
∃P(A ∪ B) ⊑ ¬D

has knowledge

has Knowledge
Definition Oriented Development

• “Reduce” (certain kinds of) effort
  – Local focus on what terms mean

• Verification
  – There are consequences to what we say
  – We can spot wrong links
    • Work on our part to detect problems
    • But inferred links are a subset of all links
  – The reasoner can tell us about broken definitions
    • We still need to understand them!

• Improve interaction
  – The KR becomes “reactive”

• Comes at a computational cost!
Reduce which effort?

- **Target:** Hierarchical controlled vocabularies
  - Aka taxonomies

- **Without** (logically encoded) definitions
  - We must **formulate** the definitions
  - We must **put terms** “in their proper place”
  - We must assert every non-trivial “link”
  - We must check that these are the **right** links
    - Thus we must determine what all the right links are
    - We must also verify that the links we include are right

- **How much work?**
  - 100 terms \(\approx 10,000 (100^2)\) possible subsumptions!
    - Plus 100 definitions
    - Link checking isn’t enough — paths!
      - Depth adds complexity
      - Multiple inheritance adds significant complexity
Types of Bicycle injury

• 1972 ICD-9 (E826) 8
• READ-2 (T30..) 81
• READ-3 87
• 1999 ICD-10 …..
1999 ICD10: 587 codes

- V31.22 Occupant of three-wheeled motor vehicle injured in collision with pedal cycle, person on outside of vehicle, nontraffic accident, while working for income
- W65.40 Drowning and submersion while in bath-tub, street and highway, while engaged in sports activity
- X35.44 Victim of volcanic eruption, street and highway, while resting, sleeping, eating or engaging in other vital activities

Taxonomic position

• Not just hard, but (perhaps) impossible
  – No one hierarchy
    • “Discussion of the hierarchies frequently will elicit comments from the domain expert about the hierarchy structure. Not infrequently in biomedicine, there is no canonical determination of a concept’s correct tree position. For example, meningococcal meningitis may be classified correctly as both a disease of the central nervous systems and a bacterial disease.” — Modeling a description logic vocabulary for cancer research
  – Hierarchies aren’t neutral!
    • “In meta-utopia, the lab-coated guardians of epistemology sit down and rationally map out a hierarchy of ideas...This presumes that there is a "correct" way of categorizing ideas, and that reasonable people, given enough time and incentive, can agree on the proper means for building a hierarchy. Nothing could be farther from the truth. Any hierarchy of ideas necessarily implies the importance of some axes over others.” — Metacrap: Putting the torch to seven straw-men of the meta-utopia
    • ??
Lab Coat View

Nothing:
Black holes

Everything:
Matter:
Earth:
Planets
Washing Machines
Wind:
Oxygen
Poo-gas
Fire:
Nuclear fission
Nuclear fusion
"Mean Devil Woman" Louisiana Hot-Sauce

Mary Van Rensselaer Buell (1893-1969)
http://www.flickr.com/photos/smithsonian/3322785642/

http://www.well.com/~doctorow/metacrap.htm#2.5
Duelling Manufacturers

Energy consumption: vs Water consumption:
Size: Capacity: Reliability

Color: Size: Programmability: Reliability

Energy
- Manufacturer
- Model
- More efficient
  - A
  - B
  - C
  - D
  - E
  - F
  - G

Less efficient
- Energy consumption kWh/cycle
  - 0.95
- Washing performance
  - A: higher, G: lower
- Spin drying performance
  - A: higher, G: lower
  - Spin speed (rpm)
  - 1400
- Capacity (cotton) kg
  - 5.0
  - 55
- Water consumption l
- Noise (dB(A) re 1 pW)
  - Washing: 5.2
  - Spinning: 7.0

Further information is continued in product brochures.
Reconciling?

• “For example, meningococcal meningitis may be classified correctly as both a disease of the central nervous systems and a bacterial disease. So there are always things the experts will question. These discussions of why the hierarchies are structured as they are offer the opportunity to introduce the notions of roles, since the hierarchy position of defined concepts are the result of the concept’s role restrictions.”

• Here, the definitions are helping conceptualization
  – Perhaps the reasoner isn’t needed per se!
Coordination

• “...one key problem limiting re-use and application-independence is that the formalisms used force application-specific choices...one of the key application-specific decisions which formalisms force on users is the organisation of a fixed taxonomy based on indivisible representations of concepts. Inevitably, indivisible representations of complex concepts conflate several ideas, and since any fixed organisation of such complex ideas is fundamentally arbitrary, choices can only be made on the basis of the current application...avoiding such decisions is an important technique for achieving reuse.
Coordination Example

• Imagine a simple taxonomy
  – Nuts and nut allergy

• Notice the lack of specificity
  – No PecanAllergy, HazelnutAllergy, etc.
  – Why not?

• Consider a form using this
  – Very simple
  – We have a patient with an almond allergy
    • Do we select
      – Allergy?
Coordination Example

- Imagine a simple taxonomy
  - Nuts and nut allergy
- Notice the lack of specificity
  - No PecanAllergy, HazelnutAllergy, etc.
  - Why not?
- Consider a form using this
  - Very simple
  - We have a patient with an almond allergy
    - Do we select
      - Allergy?
      - Nut Allergy?
Coordination Example

• Imagine a simple taxonomy
  – Nuts and nut allergy

• Notice the lack of specificity
  – No PecanAllergy, HazelnutAllergy, etc.
  – Why not?

• Consider a form using this
  – Very simple
  – We have a patient with an almond allergy
    • Do we select
      – Allergy?
      – Nut Allergy?
      – Other?
Coordination Example

• All choices have problems
  – The fact of (divergent) choice
  – The uncontrolled bits
• Can definitions help?
Coordination by definition

- Note that now we have some expressive power
Coordination by definition

- UI might be tricky...

Condition code: [blank]
Condition expression:
- Allergy that caused by some Almond

Condition code: Allergy
Condition expression:
- causedBy some Almond

Condition code: NutAllergy
Caused by:
- Nut
- Almond
- Hazelnut
- Pecan
- Walnut
Runtime vs. Development Time

• We can see benefits at each stage
• Each stage has different requirements
  – E.g., Runtime has higher performance demands
  – E.g., Development time has higher correctness demands
• More on this to come!