So many Logics, so little time

- One classical way to organize them is by tractability vs. expressivity.
- Description Logics (DLs) are basically subsets of FO logic, located in a 'sweet spot' which provides reasonable expressivity while giving *guaranteed* tractability.
- Which is good.

Safety

- The basic methodology of DL (and rule) design is, to keep things *safe*.
- Like safety wheels on a child's bicycle: you *can't possibly* fall over.
- This is done by imposing syntactic restrictions on what can be said. Then the *parser* is the safety wheel.
- It is quite tricky to design things to be safe, and the restrictions get quite complicated and arbitrary seeming. No uncles in OWL.

SAFETY vs. EXPRESSIVITY

- What happens in practice is, people want to say things which go beyond the design, and try hard to find ways to do that. Sometimes, try *very* hard.
- all men like all cats
- all men like all cats?
- The result is a kind of running battle between the users, who want to say more things, and the logic developers, who want to *prevent* them saying those things, in case that will break the worst-case performance guarantee of their reasoners.

Meanwhile

- At the other extreme, many logics are *extensions* of FO logic designed expressly to *add* expressivity which is felt to be missing.
- Modal logic, higher-order logic, hybrid logic, reflexive logic, context logic, temporal logic, ...
- None (well, maybe only a few) of these are really necessary. *First-order logic is all the logic you need.*

First-order logic is all the logic you need.

- Provided you get the logic right, and provided you don't try to make the logic do things that no mere logic can possibly do. Like *naming*.

Getting First Order right

- It does *not* mean you can't quantify over higher-order things.
- It does *not* mean you have to keep individuals and relationships completely separate, or use different vocabularies.
- These are first-order:
- (forall (y)(if (Human y) (exists (x)(and (x y)(Nationality x)))))
- (Set Set)

Getting First Order right

- First-order means being *existentially neutral*. The *only* assumption that the logic itself makes about what exists, is that every name denotes something.
- All other logics come with some kind of built-in special existence assumptions.
- That is *all there is* to being first-order.

Getting First Order right

- If we remove all the other traditional baggage of FOL, and clean away all other restrictions, we get ISO Common Logic.
- Just one kind of name, can appear in any position, can always be quantified. No restrictions on syntax at all.
- So it is fine to write apparently 'higher-order' things such as
- (iff (TransitiveRelation r)(forall (x y z)(if (and (r x y)(r y z))(r x z))))

DONGLES

- Quite a lot of extant formalisms have syntax dongles, which carry no meaning but are there only to make the syntax look 'right'.

Pat is a Brit
Brit is a national category
(Brit Pat)
(NationalCategory Brit)

Pat rdf:type Brit
Instance Brit Pat
hasType Pat Brit
AppliesTo Brit Pat
isa Pat Brit
Pat a Brit

- Most of these are needed to keep the property (class, predicate, relation) in the same syntax class as the individual. But CL has no syntax classes, so these are all unnecessary.

Atomic sentences

- All atomic sentences have a single form: (relation <sequence of arguments>)
- The commonest case is a sequence of two arguments.
- If the sequence has a single item then the relation is a *class*: (Human Pat)
- But there is also the zero case, of *no* arguments: (Foo). Foo here is a thing which yields a single truth-value. It can be thought of as a *proposition*. Propositions come for free, if we allow this syntax. (Observation due to Chris Menzel.)
- Allowing the zero case turns out to be a powerful heuristic. They are always useful, and sometimes provide important insights.

Bad philosophy

Things change as time goes by. How do we describe this? One view (OBO) is that we must distinguish between *occurrents*, which happen, and *continuants*, which simply exist, retaining their identity, as time passes. **We must make this distinction because the logic requires it.**

Occurrents have phases, we can speak of their beginnings and ends. They naturally have a temporal parameter. But continuants cannot possibly have a temporal parameter, because they are the *same* at *every* time. So we must make the distinction, because each kind of thing has its own special way to be described: (relation c t) for continuants and (relation (o t)) for occurrents.

Other ontological frameworks vehemently reject this distinction and treat everything as having temporal parts.

But in fact it is easy to relate these styles of description, if we have the right kind of syntactic freedom.

(forall ((t time)(c Continuant) relation)(iff (relation c t)(relation (c t))))

So we can let a thousand flowers bloom together.

Bad philosophy (aside)

- Why so much energy and passion?
- **Step away from the philosophy and keep your hands in the air.**
- Think of it purely syntactically. The temporal parameter has to go *somewhere*. If we can write axioms relating the various cases (we can) then *it really doesn't matter* which one you choose. In fact, there's a fairly simple algorithm which can compute the most general unifier of expressions written with the temporal parameter in *any* location.
- At this point, ontological *engineering* is done. Any other debates are ontological *philosophy*.

Logic can't fix the meanings of names.

- ".... provided you don't try to make the logic do things that no mere logic can possibly do. Like *naming*."
- What does '*Everest*' refer to? Or '*Paris*'? Or '*Barack Obama*'? How do names of things get "attached" to their referents?
- Not by logic, for sure. (Because of Herbrand's theorem.)

Logic can't fix the meanings of names.

- Even applied to purely mathematical identifiers, logical axioms can't hack it (Goedel). And yet we do manage to do arithmetic.
- What logical descriptions can do is *constrain* the *possible* meanings of names. But the actual grounding of reference has to be done some other way.
- For now, that can be the way the Caterpillar did it.

Kinds of names

- Strings '3ad4s&'
- Datatyped strings
- ('326' xsd:integer) ('02062011' xsd:date)
- Proposition names (IKL)
- (that (exists (x)(and (Human x)(Name x 'Pat'))))
- In general, (that <sentence>) is the proposition which is true just when the <sentence> is true.

Proposition names are eliminable

IKL is reducible to Common Logic, because the proposition names are eliminable by a 'skolemization' transformation.

<IKLsentence1> contains a propositional name (that <sentence2>) with free names n1...nk maps to a conjunction

(and <IKLsentence1> [(that <sentence2>) / (F n1 ... nk)]
        (forall (n1 ... nk)(iff (F n1 ... nk) <sentence2>)))

For example

(Believes John (that (exists (x)(and (loves Harry x)(loves Bill x)))))

(and (Believes John (F Harry Bill))
      (forall (y z)(iff (F y z)(exists (x)(and (loves y x)(loves z x))))))

Moral

- FO logic, done right, is all the logic you will ever need.
- But you need something else as well. You need names for things.