Specialist Inference

Motivating Specialist Inference

Reasoning purely with axiomatic theories can be very inefficient.

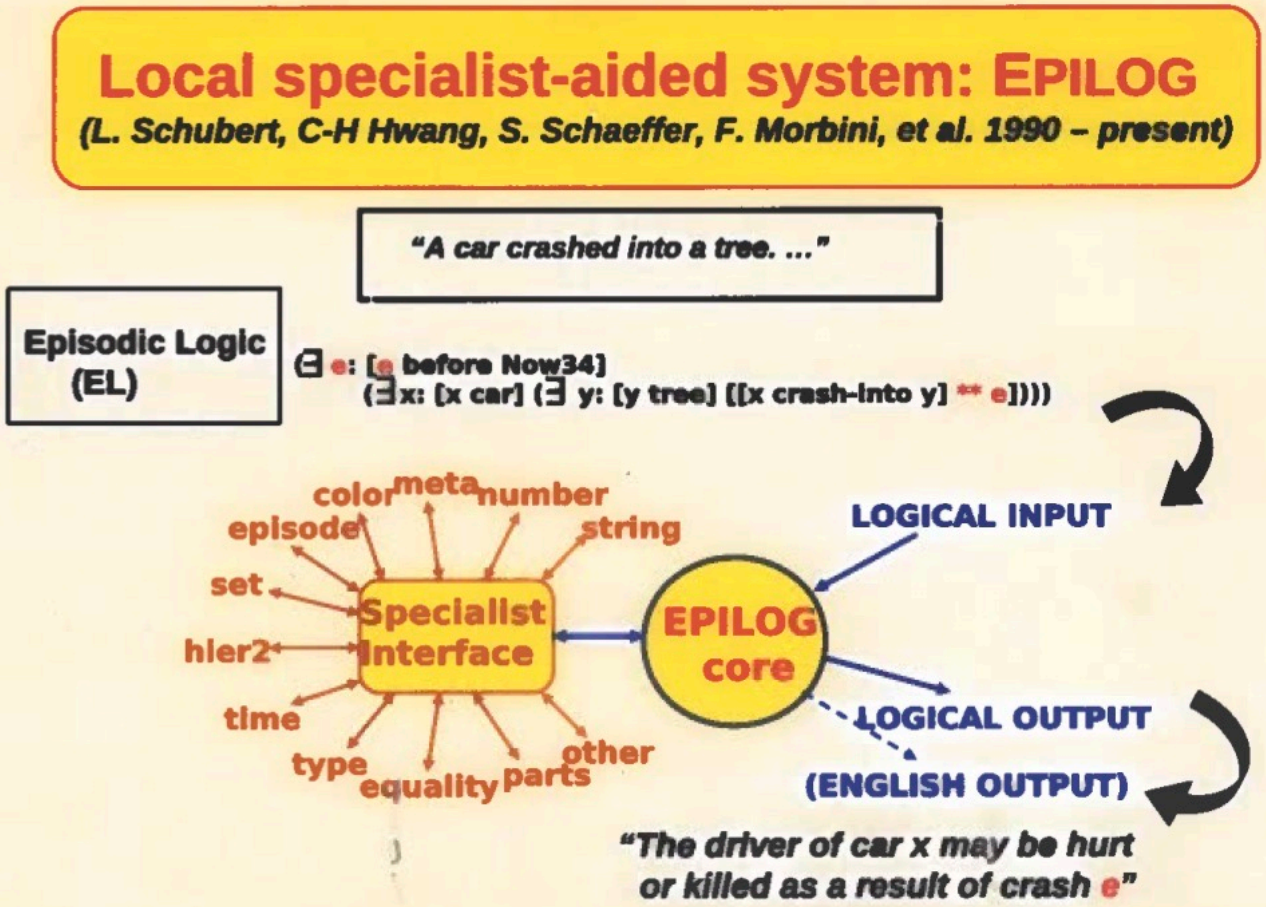
- E.g., reasoning about equality using reflexivity, symmetry, and transitivity axioms can be awkward – using paramodulation gains efficiency by "proceduralizing" this axiomatic knowledge.
- E.g., reasoning about arithmetic relations, such as that $1000+500 = 1500$, could take many steps (perhaps 500!) in an arithmetic logic
- E.g., "Whom does Mary love?" cannot properly be answered with "her prize orchid", because an orchid is not a person. This "obvious inference" might require multiple steps in an axiomatic system:

  person $\rightarrow$ creature,

  orchid $\rightarrow$ flower $\rightarrow$ plant $\rightarrow$ *not* creature.

Procedural Attachment

We can speed up such reasoning by "attaching" efficient specialized procedures to certain functions and predicates, such as +, =, <, *Before, Part-of, Person, Orchid, ...*



Overview

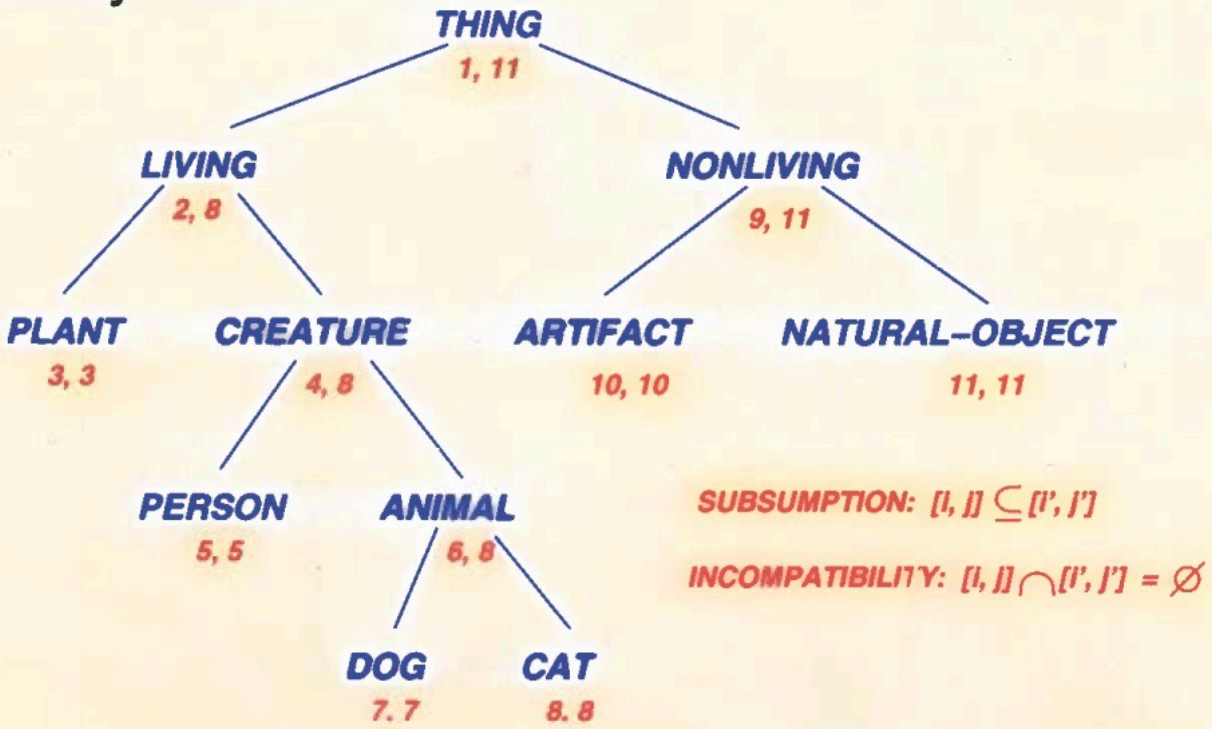
We'll look at:

- How we can design some important specialists
- How we can exploit specialized methods for *single literals*
- How we can exploit specialized methods for *multiple literals*
- Some general approaches to incorporating specialists

Some Techniques for Building Specialists

- Taxonomic Specialist (Similarly, Parts Specialist)

  Preorder node numbering allows constant-time subsumption/ compatibility checks



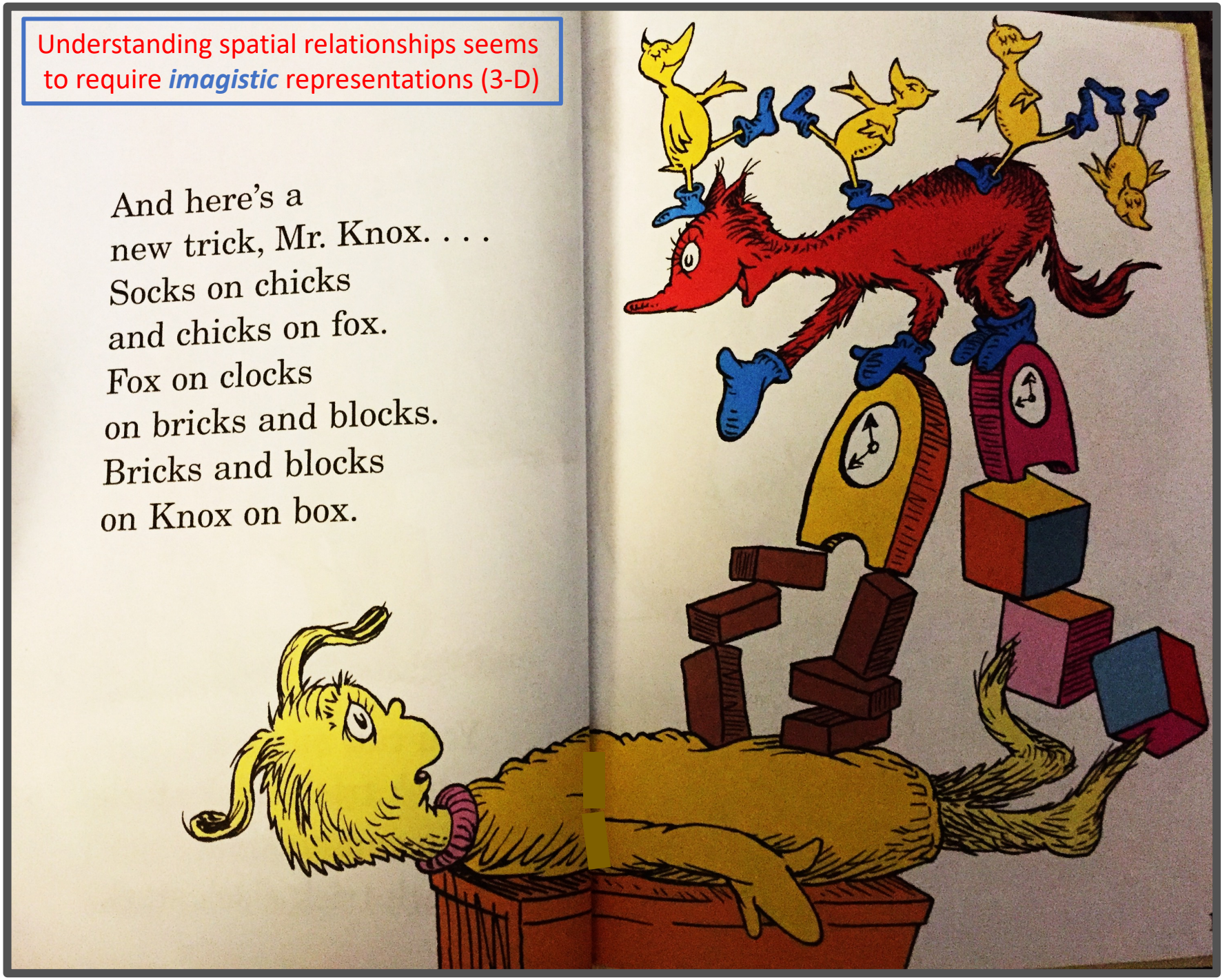
DILBERT SCOTT ADAMS

Reasoning about parts

RUBES LEIGH RUBIN

"Uh, gee, what a lovely pullover, Grandma. You shouldn't have ... really."

Understanding spatial relationships seems to require ***imagistic*** representations (3-D)

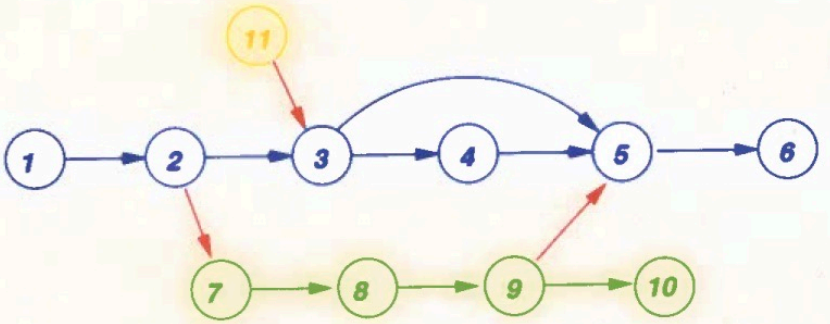


- Time Specialist

  E.g., use constraint graph for Allen Interval Algebra

  Int1 [before $\vee$ meets $\vee$ overlaps] Int_2, etc.

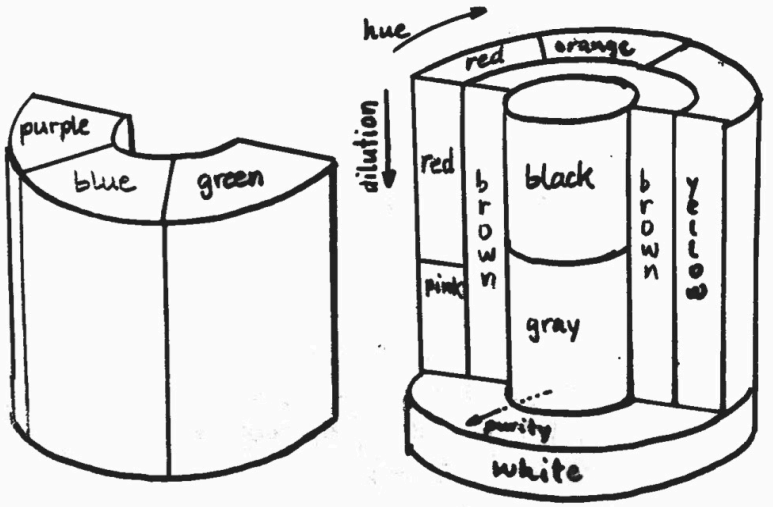
  E.g., use Timegraph for point relations



On a *chain*, checking $t_1 \leq t_2$ is *constant-time*.

SPECIALIST CAPABILITIES, cont'd

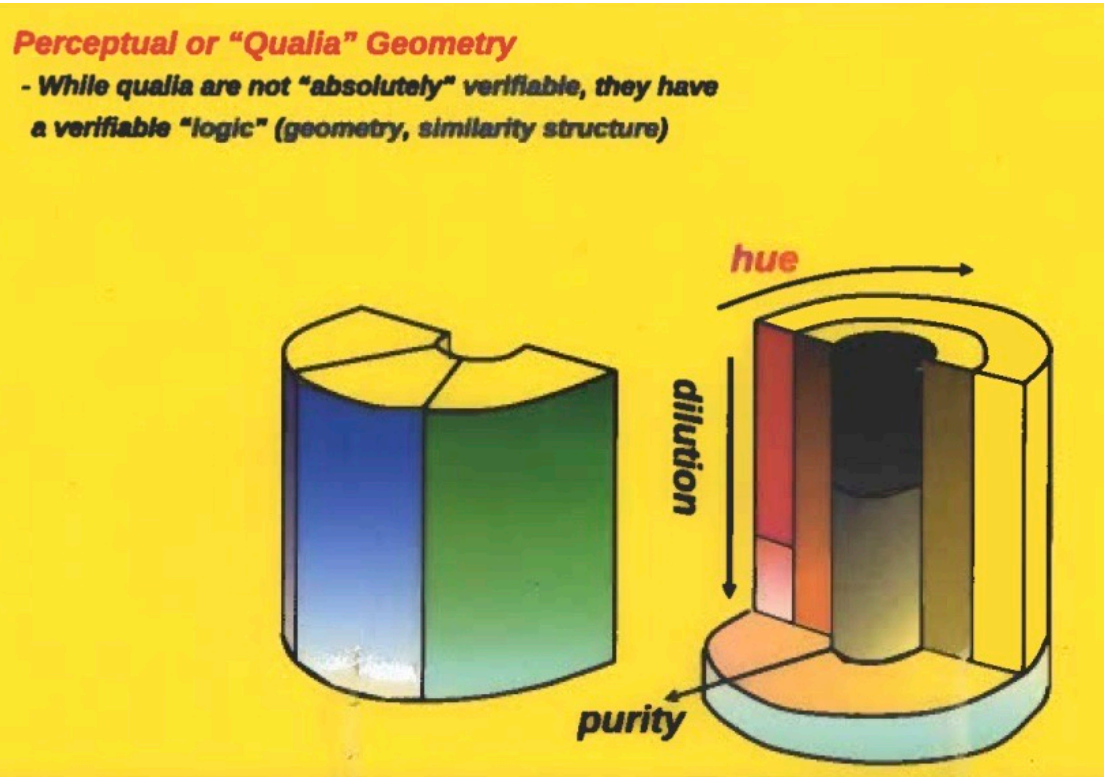
COLOR SPECIALIST



purity = pure color / (pure color + black)
dilution = white / (pure color + black + white)

- An extremely simple 3D color space, in which all the "ordinary" color terms are bounded by coordinate planes, with a natural warm/cool boundary
- constant-time compatibility/incompatibility checks, including for "hedged" terms such as blue-green, sort of pink, ...

...etc.!



Applying Specialized Methods to Single Literals

Apply these before storing any input wff or derived wff.

1. Term Simplification

   e.g., Arithmetic specialist

   $(1000 + 500) \rightsquigarrow 1500$

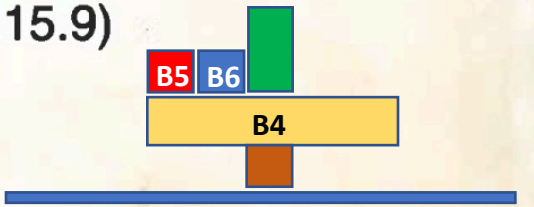
   $(C + 8 - (C - x - 7)) \rightsquigarrow (x + 15)$

   e.g., Geometry/ Physics Specialist (in blocks world, etc.)

   (weight(B5)*dist(B5,x-coord(center(B4))) + 15.9)

   $\rightsquigarrow$ 44.2

part of a stability calculation



2. False Literal Elimination

e.g., Arithmetic specialist

$(x + 8 < x - 7) \rightsquigarrow \square$

$\underline{(x + 8 < x - 7)} \vee P(x, A) \rightsquigarrow P(x, A)$

e.g., Taxonomic (Type) Specialist

Person(Orchid34) $\rightsquigarrow \square$

Person(Orchid34) $\vee$ Q(x, Mary) $\rightsquigarrow$ Q(x, Mary)

e.g., Time Specialist

Before(Shuttle-Launch1, Moon-Walk1) $\rightsquigarrow \square$

3. True Literal Simplification

e.g., Arithmetic specialist

$$\underline{\neg(x+8<x-7)} \vee P(x,A) \rightsquigarrow \neg(x+8<x-7)$$

e.g., Taxonomic (Type) Specialist

$$\underline{\text{Person(John)}} \vee \text{Q(x, Mary)} \rightsquigarrow \text{Person(John)}$$

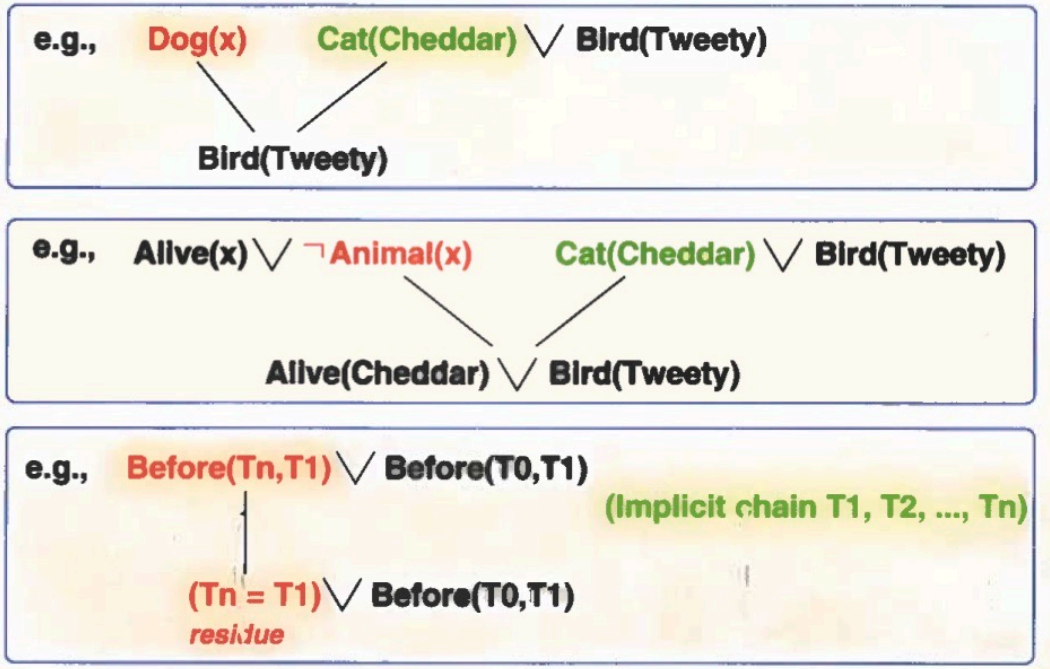
Justification: The true literal (underlined) *subsumes* the clause as a whole, so anything we can deduce with the clause we can also deduce with the true literal.

We may even be able to drop the clause altogether – not retaining the true literal – if any resolution that could be done with the true literal could instead be done by false literal elimination; e.g., we don't need Person(John) to resolve ¬Person(John) or ¬Person(x), if we can do false literal elimination for them.

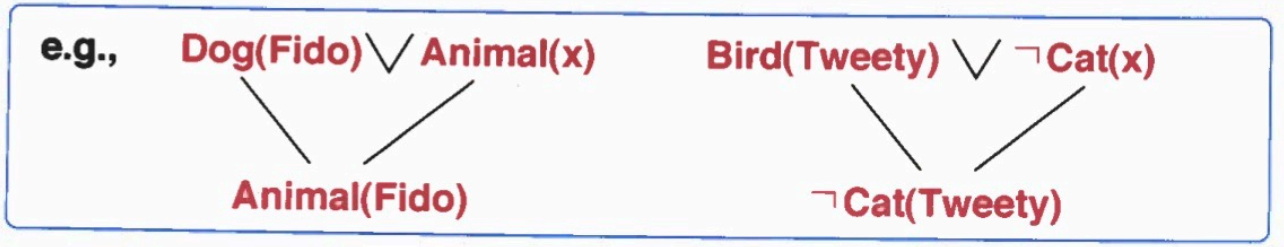
Applying Specialized Methods to Multiple Literals

1. Generalized Resolving

   e.g., Plant(Orchid34), Person(x) $\rightsquigarrow \square$



2. Generalized Factoring



3. Generalized Subsumption Elimination

e.g., Cat(f(x)) subsumes

Creature(f(C)) as well as ¬Artifact(f(C))

[The remaining slides are graduate-level material, and are only sketchily done here.]

4. Theory Resolution (M. Stickel 1983)

Generalizes "false literal elimination", "generalized resolving"

(a) Select one or more literals from one or more clauses, using the implicit theory to form their "residue" (often = $\square$), and obtaining unifier γ.

(b) Infer the disjunction of remaining literals under substitution γ, also disjoining the residue (if $\neq \square$). This is the *theory resolvent*.

For completeness, the residue must be "the strongest consequence" of the resolved literals: it must be inconsistent (relative to the implicit theory) with any set of clauses that the resolved literals were inconsistent with (relative to the implicit theory).

Also, some (ordinary or theory) resolution step must be possible for any set of clauses that are unsatisfiable (relative to the implicit theory).

Two other related techniques

- **Sortal Resolution**
  - Divide predicates into *sortal* and *nonsortal*
  - Devise special methods for testing compatibility of sortal literals
  - Write sortal literals separately as "constraints" on variables

    e.g., $\neg$Dog(x) $\vee \neg$Cat(y) $\vee$ Hates(x,y) $\vee$ Afraid-of(x,y)

    becomes

    Hates(x,y) $\vee$ Afraid-of(x,y) / Dog(x), Cat(y)

  Sorted clauses may be resolved only if their sortal constraints are compatible, according to the implicit theory. This can be very efficient, though it is less general than theory resolution.

- Constraint Logic Programming
  - Uses sortal resolution as a built-in programming language feature
  - Sortal constraints are additional conditions on the RHS of a Horn clause
  - Potentially allows multiple sortal theories: e.g., taxonomies, inequalities