Classification & Experimentation

Bijan Parsia
<bijan.parsia@manchester.ac.uk>

Today

- Some brief hints on engineering
- Thinking about the problem
- Classification without heat death
- Experimentation on reasoning

A Tale of Three Rules

Wherein we see that a little thought goes a long way

Consider the ⊓-rule

⊓-rule: if $C_1 \sqcap C_2 \in L(a)$ for some a and $\{C_1, C_2\} \not\subseteq L(a)$

then add $\{C_1, C_2\}$ to $L(a)$

- How do you implement this?
- When do you fire this?

Consider the clash rule

$$\{\mathsf{A}, \neg\mathsf{A}\} \not\subseteq L(a),\ \bot \notin L(a) \text{ for all } a, \mathsf{A}$$

- How do you test for clashes?
- How often do you test for clashes?

Consider the $\sqcup$-rule

$\sqcup$-rule: if $C_1 \sqcup C_2 \in L(a)$ for some a and $\{C_1, C_2\} \cap L(a) = \emptyset$

then replace $\mathcal{G}$ with $\mathcal{G}_1$ and $\mathcal{G}_2$ s.t.

$C_1 \in L(a)$ in $\mathcal{G}_1$ and $C_2 \in L_2(a)$ in $\mathcal{G}_2$

- How do you create G1 and G2?
  - Deep copy?

There's other stuff in your code!

```
debugString = currentClass.toString();
if (debug)
   System.out.println(debugString);
```

```
debugString = currentClass.toString();
if (debug)
   System.out.println(debugString);
```

- What's wrong with this?
- What should it look like?
- This appeared in real code
  - by one of the top 4 programmers I know

Classify Classification

Wherein we despair

Classification: Formulations

- Most standard formulation:
  - For all class names A, B in $\mathcal{O}$, determine whether
    - $\mathcal{O} \models A \sqsubseteq B$ (or not)
- Slight generalisation
  - For all class names (plus $\bot$ and $\top$), A, B in $\mathcal{O}$, determine whether
    - $\mathcal{O} \models A \sqsubseteq B$
  - This neatly includes
    - the consistency check (i.e., whether $\mathcal{O} \models \top \sqsubseteq \bot$)
    - concept satisfiabilities (i.e., whether $\mathcal{O} \models A \sqsubseteq \bot$)
    - concept trivialities (i.e., whether $\mathcal{O} \models \top \sqsubseteq A$)
- Full generalisation
  - For all predicate names (user defined or built in) A, B in $\mathcal{O}$, determine whether
    - $\mathcal{O} \models A \sqsubseteq B$
  - This includes roles/properties/binary predicates!
  - Few systems do this! (Only HermiT?)

Classification: For implementation (1)

- Close Functional Formulation (CFF)
  - Input: An ontology $\mathcal{O}$ (i.e., a set of axioms)
  - Output: A new ontology $\mathcal{O}'$ st
    - $\mathcal{O}' = \{A \sqsubseteq B \mid A, B \in (\tilde{\mathcal{O}} \cup \{\bot, \top\}) \;\&\; \mathcal{O} \models A \sqsubseteq B\}$ or
      - Transitive closure
    - $\mathcal{O}' = \{A \sqsubseteq B \mid A, B \in (\tilde{\mathcal{O}} \cup \{\bot, \top\}) \;\&\; \mathcal{O} \models A \sqsubseteq B \;\&\; \nexists C \text{ s.t. } \{A \sqsubseteq C, C \sqsubseteq B\} \subseteq \mathcal{O}'\}$
      - Transitive reduct...almost
    - (Pick your favorite data structure to represent this)
- Problems?

CFF Problems: ⊥ and ⊤

- Many subsumptions involving ⊥ and ⊤ are trivial
  - A ⊑ ⊤
  - ⊥ ⊑ A
  - Other key trivial subsumption:
    - A ⊑ A (and circular!)

A ⊑ ⊤
C ⊑ A

vs.

A ⊑ ⊤
C ⊑ ⊤

- Non trivial examples “blow up” the transitive closure
  - If A ⊑ ⊤, then A is subsumed by every other term

A ⊑ B
C ⊑ D

A ⊑ B	⊥ ⊑ A
C ⊑ D	⊥ ⊑ B
A ⊑ ⊤	⊥ ⊑ C
B ⊑ ⊤	⊥ ⊑ D
C ⊑ ⊤	
D ⊑ ⊤	

A ⊑ B
C ⊑ D
A ⊑ ⊥

A ⊑ B	A ⊑ ⊤	⊥ ⊑ A
C ⊑ D	B ⊑ ⊤	⊥ ⊑ B
A ⊑ ⊥	C ⊑ ⊤	⊥ ⊑ C
A ⊑ C	D ⊑ ⊤	⊥ ⊑ D
A ⊑ D		

CFF Problems: Equivalences

- In CFF, if A ≡ B, then
  - it shows up as A ⊑ B and B ⊑ A
- But what happens with longer chains?
- Problems even
  - if we allow equivalences

$$\boxed{A \equiv B}$$

$$\boxed{\begin{array}{l} A \equiv B \\ B \equiv C \end{array}}$$

$$\boxed{\begin{array}{l} A \equiv B \\ B \equiv C \\ A \equiv D \end{array}}$$

$$\boxed{\begin{array}{l} A \sqsubseteq B \\ A \sqsubseteq C \\ A \sqsubseteq D \\ B \sqsubseteq A \\ B \sqsubseteq C \\ B \sqsubseteq D \\ C \sqsubseteq A \\ C \sqsubseteq B \\ C \sqsubseteq D \\ D \sqsubseteq A \\ D \sqsubseteq B \\ D \sqsubseteq C \end{array}}$$

$$\boxed{\begin{array}{l} A \equiv B \\ B \equiv C \end{array}} \text{ vs. } \boxed{\begin{array}{l} A \equiv B \\ A \equiv C \end{array}}$$

$$\boxed{\begin{array}{l} A \sqsubseteq B \\ B \sqsubseteq A \end{array}}$$

$$\boxed{\begin{array}{l} A \sqsubseteq B \\ A \sqsubseteq C \\ B \sqsubseteq A \\ B \sqsubseteq C \\ C \sqsubseteq A \\ C \sqsubseteq B \end{array}}$$

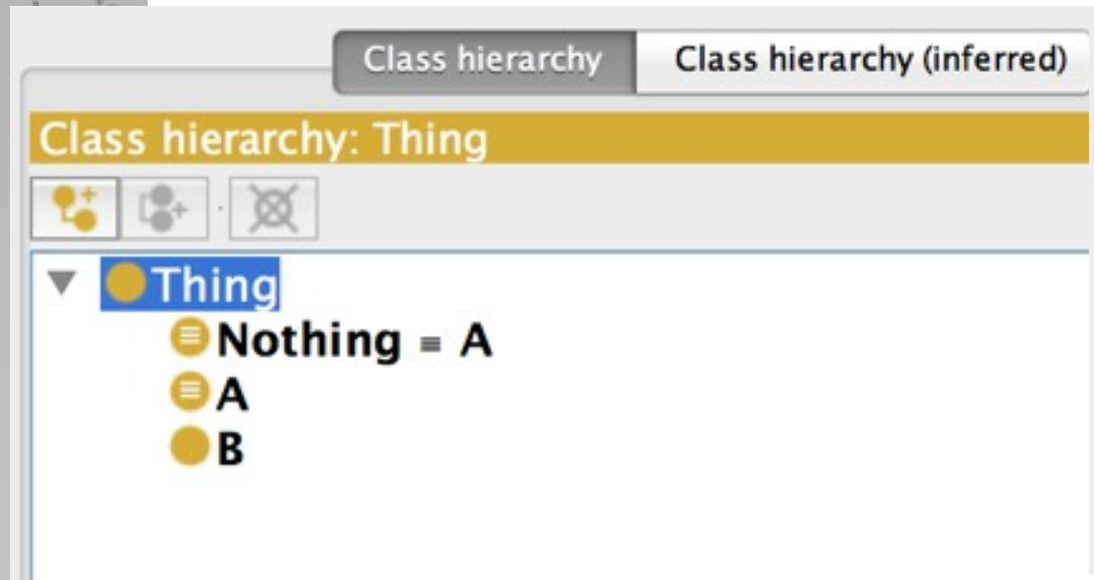
Transitive Reduct Saves the Day?

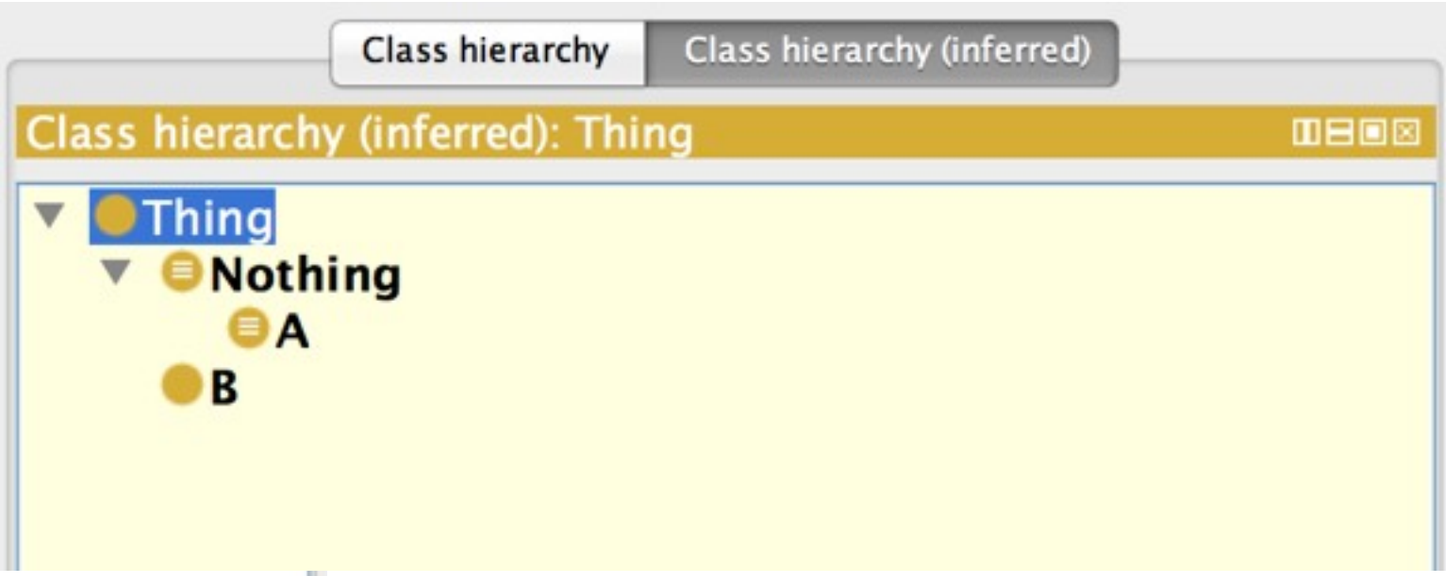
- Modified Functional Formulation (CFF)
  - Input: An ontology $\mathcal{O}$ (i.e., a set of axioms)
  - Output: A new ontology $\mathcal{O}'$ st
    - $\mathcal{O}' = \{A \sqsubseteq B \mid A, B \in (\tilde{\mathcal{O}} \cup \{\bot, \top\}) \;\&\; \mathcal{O} \models A \sqsubseteq B^*$
      $\&\; \nexists C \text{ s.t. } \{A \sqsubseteq C, C \sqsubseteq B\} \subseteq \mathcal{O}'$
      $\&\; \mathcal{O} \not\models A \equiv B\} \cup$
      $\{\equiv(A_1 \ldots A_n)^{**} \mid A_1 \ldots A_n \in (\tilde{\mathcal{O}} \cup \{\bot, \top\}) \;\&$
      $1 \le i,j \le n^*,\; \mathcal{O} \models A_i \equiv A_j\}$

    *And a few more side conditions

    **Where A1…An is “appropriately” sorted
  - (Pick your favorite data structure to represent this)
- Great for some applications
  - But the downstream application should know your particulars!
- Bad for some applications

Downstream apps...oy!





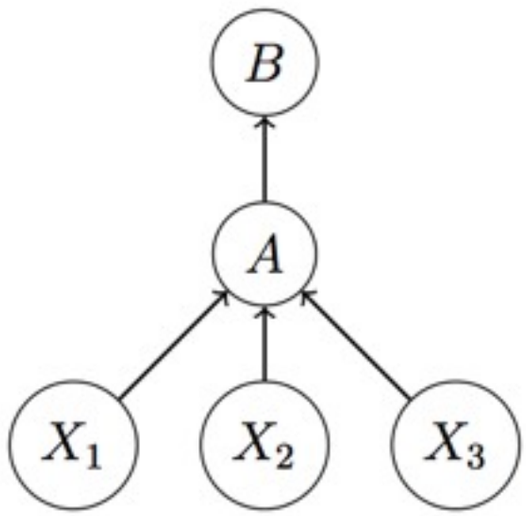

```
Declaration(Class(:A))
EquivalentClasses(:A owl:Nothing)
Declaration(Class(:B))
EquivalentClasses(owl:Nothing :A)
```

```
Declaration(Class(:A))
EquivalentClasses(:A owl:Nothing)
SubClassOf(:A owl:Thing)
Declaration(Class(:B))
SubClassOf(:B owl:Thing)
EquivalentClasses(owl:Nothing :A)
```

Counting Entailments

- Goal:
  - Given $O1$ and $O2$, determine
    - whether $O1$ has "more entailments" than $O2$
    - restrict our attention to atomic subsumptions
- Easy if one entails the other
- Transitive reduct fails to be monotonic

X1 ⊑ A
X2 ⊑ A
X3 ⊑ A
X1 ⊑ B
X2 ⊑ B
X3 ⊑ B
A ⊑ B

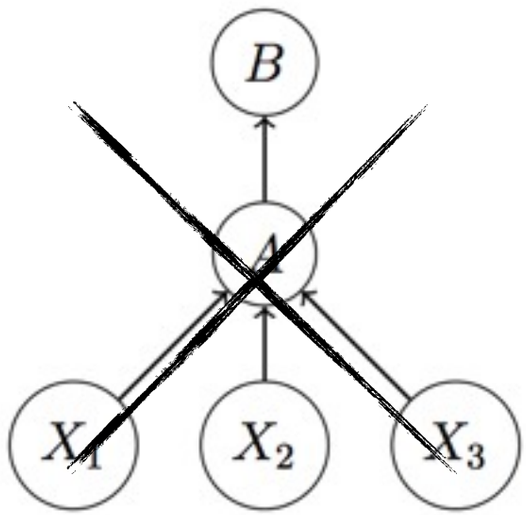


4

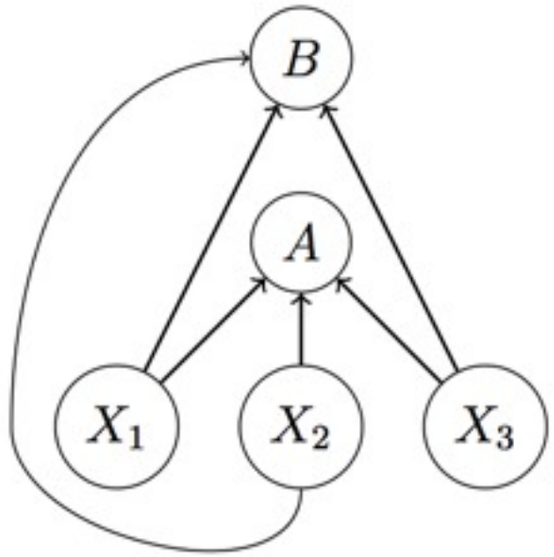
Counting Entailments

- Goal:
  - Given *O1* and *O2*, determine
    - whether *O1* has “more entailments” than *O2*
    - restrict our attention to atomic subsumptions
- Easy if one entails the other
- Transitive reduct fails to be monotonic

X1 ⊑ A
X2 ⊑ A
X3 ⊑ A
X1 ⊑ B
X2 ⊑ B
X3 ⊑ B
~~A ⊑ B~~



4



6!

Extended notions

- What about disjointnesses?
  - Negative literals as well!
    - $L = \tilde{O} \cup \{\neg A \mid A \in \tilde{O}\}$
  - $\{A \sqsubseteq B \mid A, B \in L \;\&\; O \models A \sqsubseteq B\}$
    - Note redundancy and choice!
      - $A \sqsubseteq B \Leftrightarrow \neg B \sqsubseteq \neg A$
      - $A \sqsubseteq \neg B \Leftrightarrow B \sqsubseteq \neg A$
- Beyond literals!
  - We could classify sub-expressions (Sub)
    - $A \in Sub$
      $C \sqsubseteq D \in Sub \rightarrow C,D \in Sub$
      $\neg C \in Sub \rightarrow C \in Sub$
      $C \sqcup D \in Sub \rightarrow C,D \in Sub$
      $C \sqcap D \in Sub \rightarrow C,D \in Sub$
      $\exists P.C \in Sub \rightarrow C \in Sub$
      $\forall P.C \in Sub \rightarrow C \in Sub$
  - $\{A \sqsubseteq B \mid A, B \in Sub \;\&\; O \models A \sqsubseteq B\}$

Classify before you die

Wherein we avoid work

3 RoughClassification Approaches

- Reduction to SAT
  - All tableaux & hypertableaux systems
    - Dominant, covers arbitrary languages
- Consequence based
  - Currently for fragments, esp. EL and horn-SHIQ
- Meta/Modular

Subsumption tests

- There can always be n^2 subsumption tests
  - Four possible states
    1. $\mathcal{O} \models A \sqsubseteq B$
       - In all models of O, $A^I \subseteq B^I$
    2. $\mathcal{O} \not\models A \sqsubseteq B$
       - In at least one model, $A^I \not\subseteq B^I$
    3. $\mathcal{O} \models A \sqsubseteq \neg B$
       - In all models of O, $A^I \cap B^I = \varnothing$
    4. $\mathcal{O} \models \neg(A \sqsubseteq B)$
       - In every model, $A^I \not\subseteq B^I$
- We look for 1 & 2
  - 3 and 4 entail 2
  - Handy fact!

Key issues

- There can always be n^2 subsumptions
  - Consider $\mathcal{O} \models \top \sqsubseteq \bot$!
    - But this case doesn't require n^2 tests
- 1 subsumption test
  - Can dominate
  - Easiest to see in SAT based procedures
    - If SAT is NP-hard (EXPTIME, NEXPTIME, 2NEXPTIME), then one such test can kill you
    - $A \sqcap \neg B$
  - But even with a PTIME SAT test...
    - The quadratic factor can kill you

The quadratic factor

- Consider the SNOMED CT ontology
  - contains about 300,000 terms.
- Presume the naive approach
  - Perform ≈ n^2 subsumption tests
- Let your test we wicked fast
  - 1 millisecond per test
- Classification time

  300,000 × 300,000 milliseconds

  = 25, 000 hours

  ≈ 2.8 years

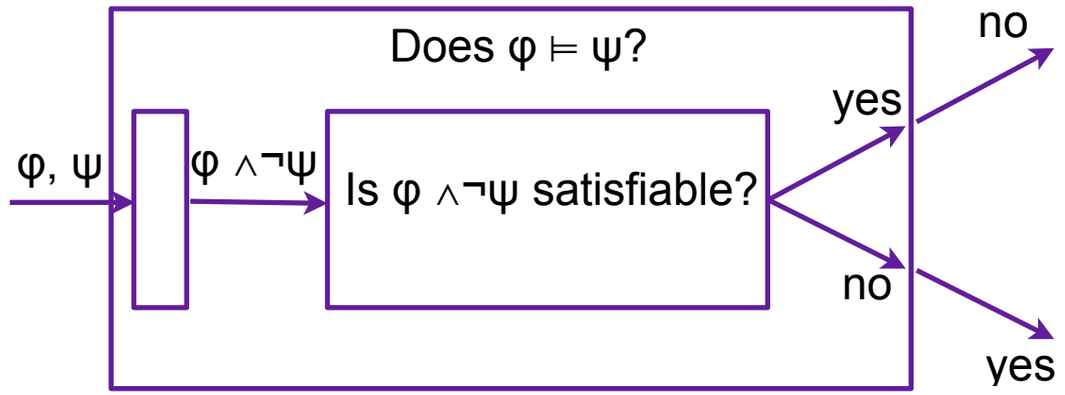
Any practically scalable classification implementation must prune the subsumption test space

SAT based procedures

Wherein we get satisfaction

SAT based procedures

- Refutation procedure
  - Via reduction to an concept
    - $C \sqcap \neg D$
- Individual SAT tests
  - Positive: Concept is unsatisfiable; subsumption holds
  - Negative: Concept is satisfiable; nonsubsumption
- Basic strategy
  1. Avoid tests
  2. Substitute cheap (generally sound, but incomplete) tests
     - SAT procedure independent (some are part of 1)
     - Exploit extra info from the SAT test
  3. Worst case, do a “full” SAT test
     - And complain about it!



Enhanced Traversal

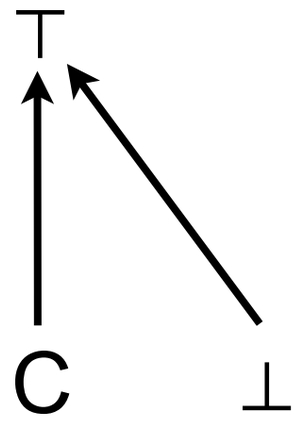
- Data structure:
  - A DAG where
    - Nodes are (sets of) concept names
    - Edges indicate subsumption relations
    - Initialize with $\bot \rightarrow \top$
- General idea
  - DAG represents the transitive reduct of atomic subsumption
  - Add subsumptions as you find them
  - Don't look for subsumptions that are
    - implicit in the graph
    - impossible in the graph
  - Defer looking for subsumptions
    - where they are unlikely

ET: Top search (Top down)

- Given a fresh concept, C, to classify
- Starting from $\top$ check whether
  - $C \sqsubseteq \top$

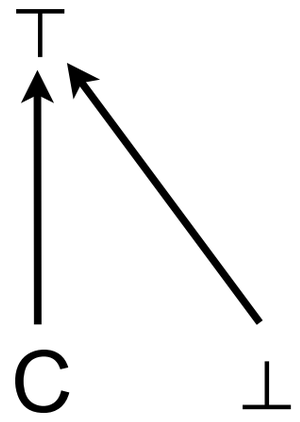
ET: Top search (Top down)

- Given a fresh concept, C, to classify
- Starting from ⊤ check whether
  - C ⊑ ⊤
    - Easy yes!
  - Only candidate left is ⊥
    - SAT test!??!?
    - (In some cases)
    - Answer (let's say): no
    - No other candidates for subsumers
    - Done Top search for C



ET: Bottom search (Bottom Up)

- Given our placed concept C
- Starting from ⊥ check whether
  - ⊥ ⊑ C

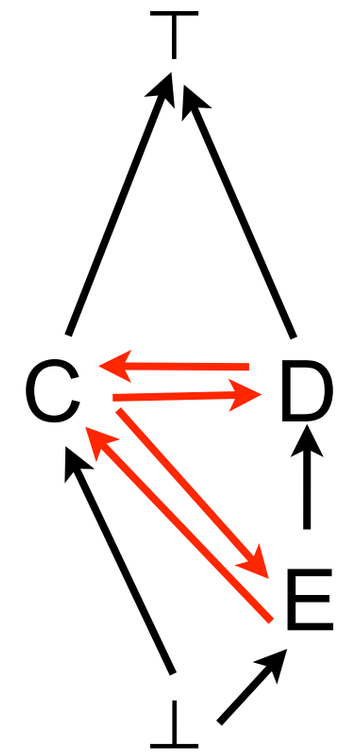


ET: Bottom search (Bottom Up)

- Given our placed concept C
- Starting from ⊥ check whether
  - ⊥ ⊑ C
    - Easy yes!
  - What's left?
    - Only candidate is ⊤
    - ⊤ subsumes all subsumees of C
      - Potential SAT test!!!
      - In this case, ⊤⋢C
      - So we're done!

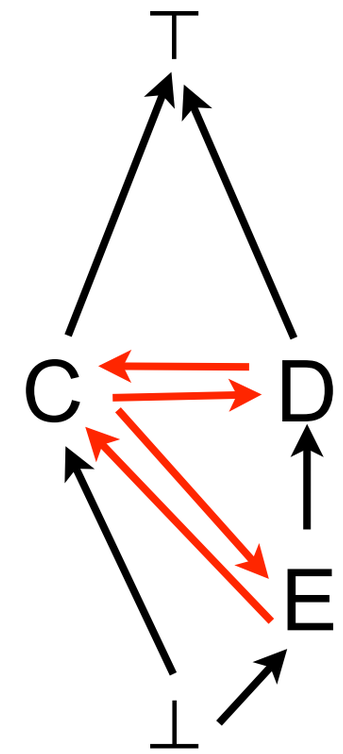
Information reuse

- Top Down
  - If we know
    - $E \sqsubseteq D$
    - $C \not\sqsubseteq D$
  - Then we know
    - $C \not\sqsubseteq E$
  - *No need to perform a test!*
- Bottom up
  - If we know
    - $E \sqsubseteq D$
    - $E \not\sqsubseteq C$
  - Then we know
    - $D \not\sqsubseteq C$
  - *No need to perform a test!*



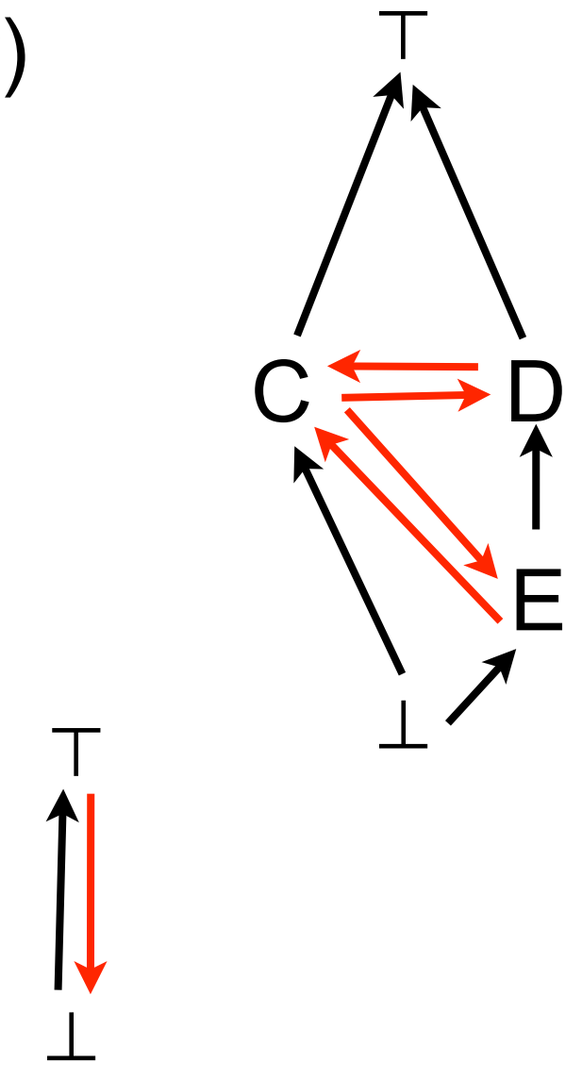
Savings

- Possible tests (assuming consistency)
  - Total
    - n = 5
    - n^2 = 25
- Count (order C, D, E)



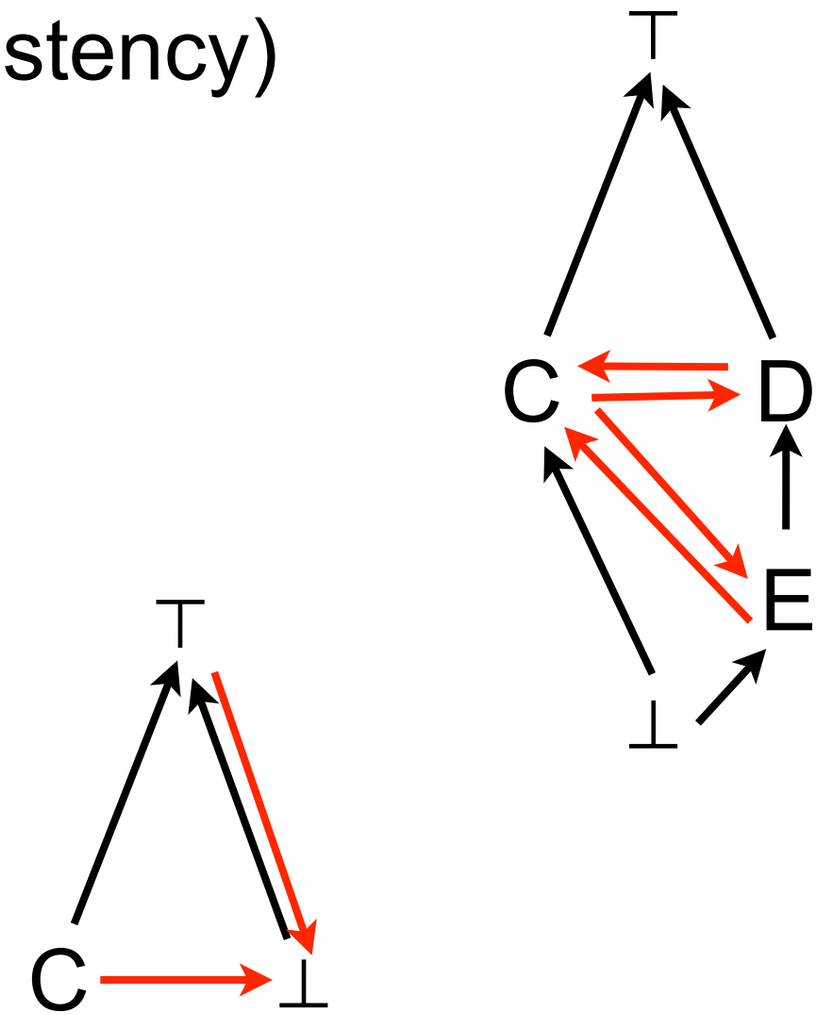
Savings

- Possible tests (assuming consistency)
  - Total
    - n = 5
    - n^2 = 25
- Count (order C, D, E)
  - (1) trivial ($\bot \sqsubseteq \top$)
  - (2) non trivial ($\top \sqsubseteq \bot$)
    - Consistent! (SAT)!
  - C
    - Top Down



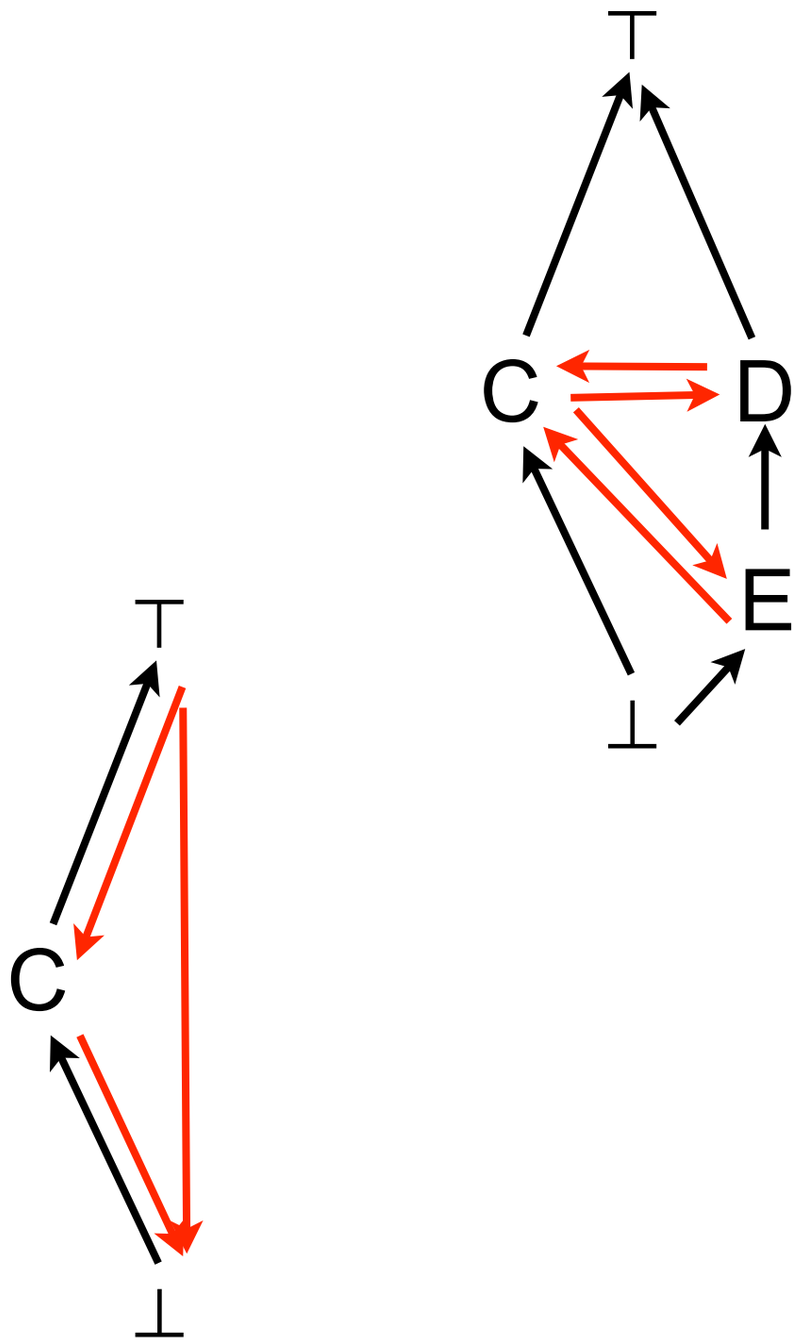
Savings

- Possible tests (assuming consistency)
  - Total
    - n = 5
    - n^2 = 25
- Count (order C, D, E)
  - (1) trivial ($\bot \sqsubseteq \top$)
  - (2) non trivial ($\top \sqsubseteq \bot$)
    - Consistent! (SAT)
  - C
    - Top Down
      - (3) C $\sqsubseteq \top$ (trivial!)
      - (4) C $\sqsubseteq \bot$ (hard!)
        - » C is is satisfiable (SAT)



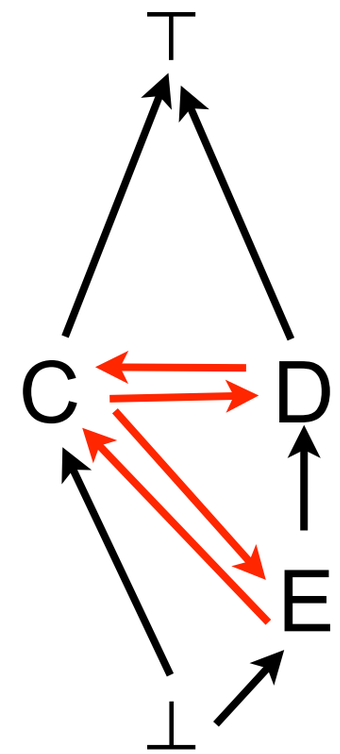
Savings

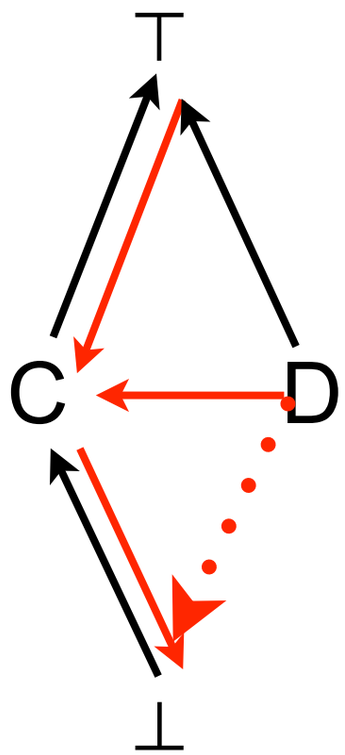
- Possible tests
  - Total
    - n = 5
    - n^2 = 25 SAT!
- Count (order C, D, E)
  - (1) trivial (⊥⊑⊤)
  - (2) non trivial (⊤⊑⊥)
    - Consistent! (SAT)
  - C [1 SAT, 1 Trivial]
    - Bottom up
      - (5) ⊥⊑C (trivial)
      - (6) ⊤⊑ C (SAT!)



Savings

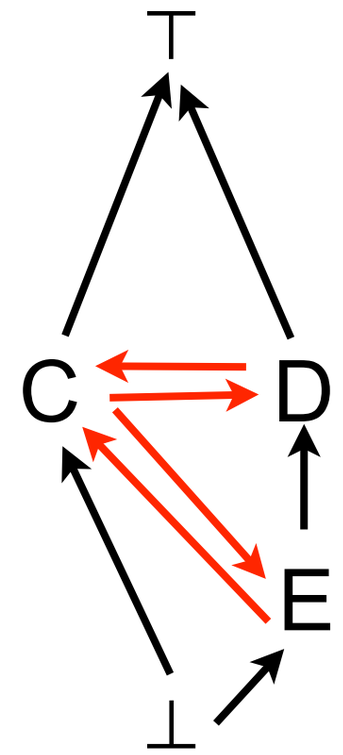
- Possible tests
  - Total
    - n = 5
    - n^2 = 25 SAT!?
- Count (order C, D, E)
  - (1) trivial ($\bot \sqsubseteq \top$)
  - (2) non trivial ($\top \sqsubseteq \bot$)
    - Consistent! (SAT)
  - C [2 SAT, 2 Trivial]
  - D
    - Top Down
      - (7) D $\sqsubseteq \top$ (trivial!)
      - (8) D $\sqsubseteq$ C (SAT!)
      - (9) D $\sqsubseteq \bot$ AVOIDED

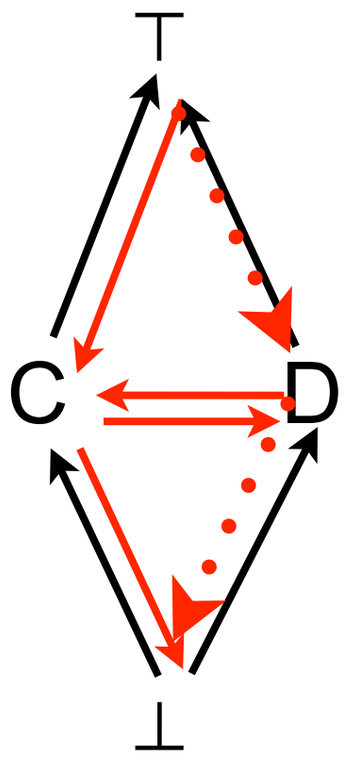




Savings

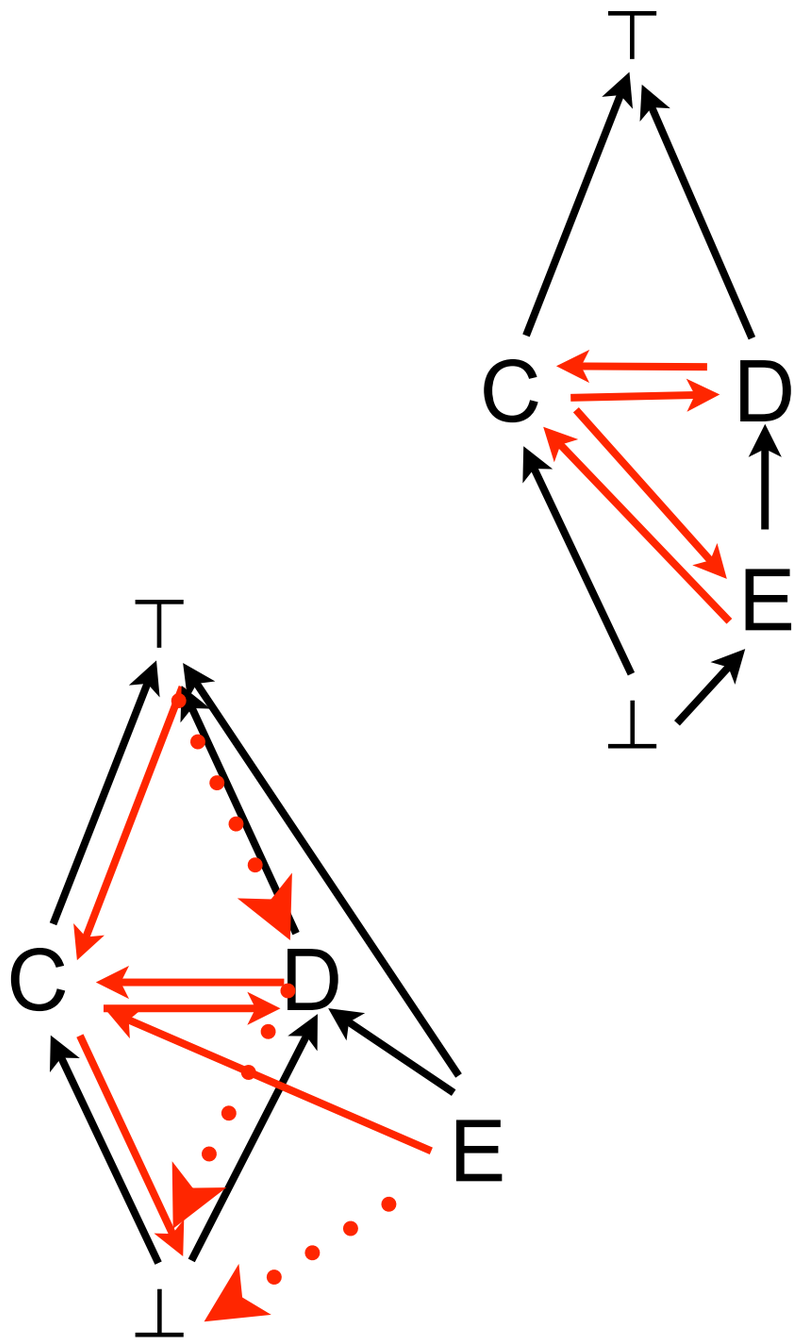
- Possible tests
  - Total
    - n = 5
    - n^2 = 25 SAT!?
- Count (order C, D, E)
  - (1) trivial (⊥⊑⊤)
  - (2) non trivial (⊤⊑⊥)
    - Consistent! (SAT)
  - C [2 SAT, 2 Trivial]
  - D [1 SAT, 1 Trivial, 1 Avoided]
    - Bottom up
      - (10) ⊥⊑ D Trivial
      - (11) C ⊑ D (SAT)
      - (12) ⊤⊑D AVOIDED!





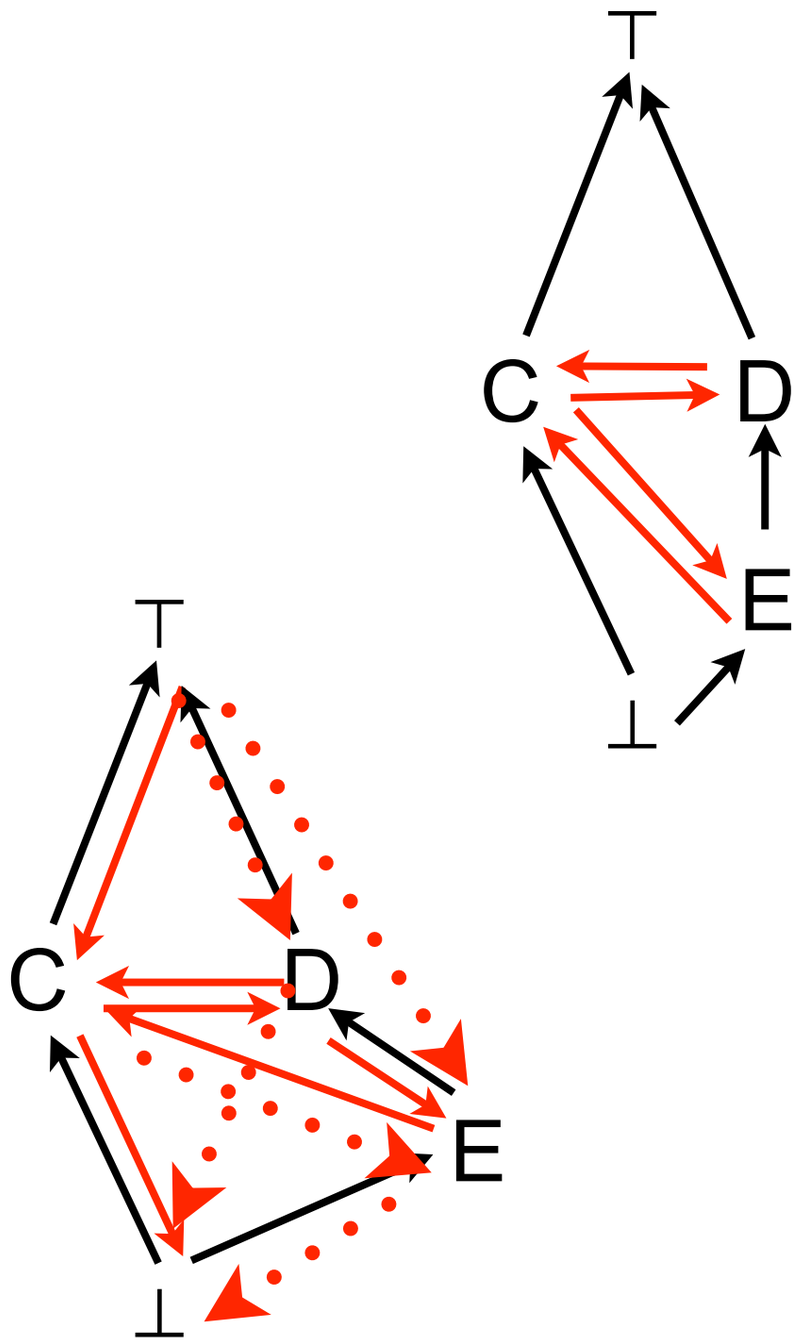
Savings

- Possible tests
  - Total
    - n = 5
    - n^2 = 25 SAT!?
- Count (order C, D, E)
  - (1) trivial ($\bot \sqsubseteq \top$)
  - (2) non trivial ($\top \sqsubseteq \bot$)
    - Consistent! (SAT)
  - C [2 SAT, 2 Trivial]
  - D [2 SAT, 2 Trivial, 2 Avoided]
  - E
    - Top Down
      - (13) $E \sqsubseteq \top$ (trivial)
      - (14) $E \sqsubseteq C$ (SAT)
      - (15) $E \sqsubseteq D$ (SAT)
      - (16) $E \sqsubseteq \bot$ (AVOIDED)



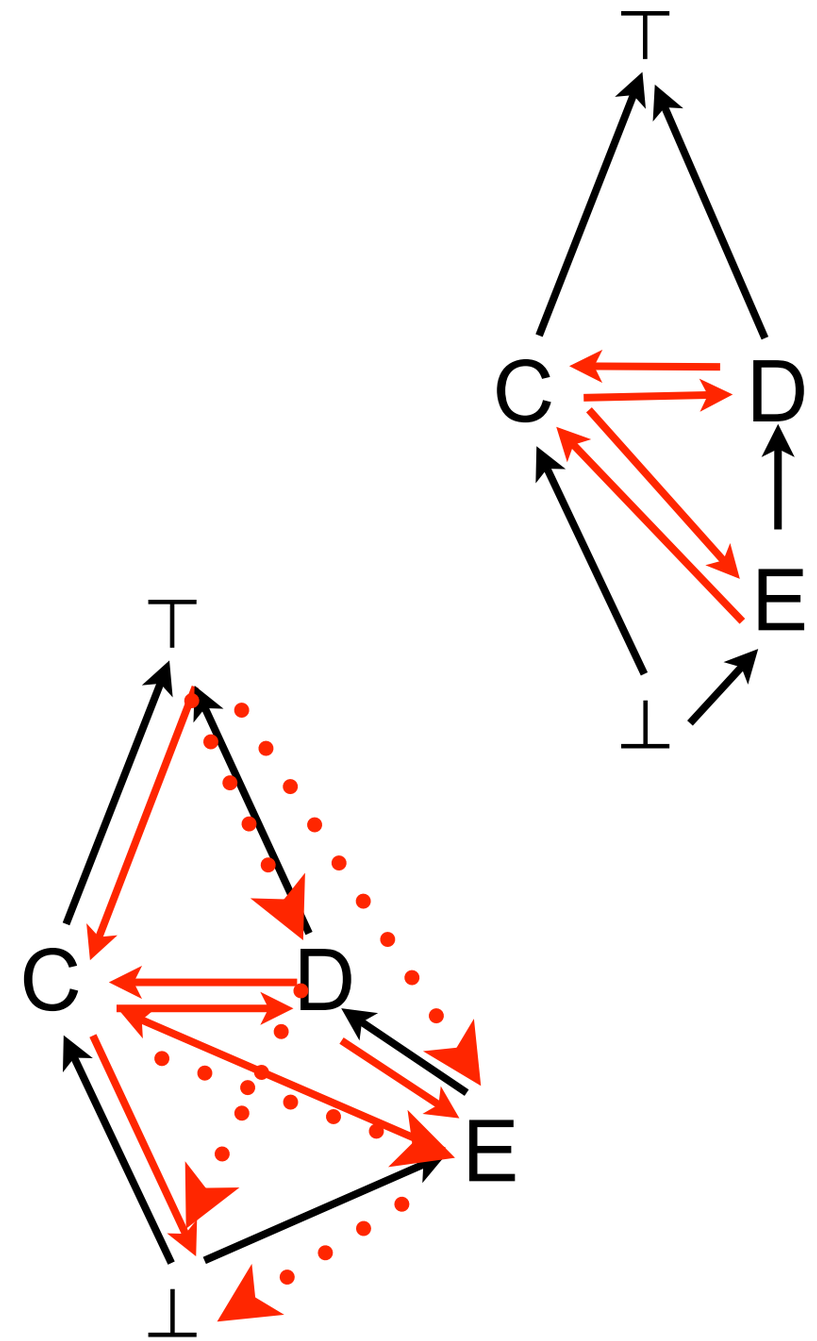
Savings

- Possible tests
  - Total
    - n = 5
    - n^2 = 25 SAT!?
- Count (order C, D, E)
  - (1) trivial ($\bot \sqsubseteq \top$)
  - (2) non trivial ($\top \sqsubseteq \bot$)
    - Consistent! (SAT)
  - C [2 SAT, 2 Trivial]
  - D [2 SAT, 2 Trivial, 2 Avoided]
  - E [1 Trivial, 2 SAT, 1 Avoided]
    - Bottom up
      - (17) $\bot \sqsubseteq$ E (trivial)
      - (18) C $\sqsubseteq$ E AVOIDED
      - (19) D $\sqsubseteq$ E (SAT)
      - (20) $\top \sqsubseteq$ E AVOIDED



Savings

- Possible tests
  - Total
    - n = 5
    - n^2 = 25 SAT!?
- Count (order C, D, E)
  - (1) trivial ($\bot \sqsubseteq \top$)
  - (2) non trivial ($\top \sqsubseteq \bot$)
    - Consistent! (SAT)
  - C [2 SAT, 2 Trivial]
  - D [2 SAT, 2 Trivial, 2 Avoided]
  - E [2 Trivial, 3 SAT, 3 Avoided]
  - Reflexive!
    - $C \sqsubseteq C$
      - All avoided = 5
- Total [8 SAT, 7 Trivial, 5+5 avoided] = 25



9 SAT seem like a lot!

- Assertions!
  - If our ontology contains $E \sqsubseteq D$
    - We can just enter that! No sat!
  - If our ontology contains (or implies) $C \sqsubseteq \neg D$
    - Then we don't need to test $D \sqsubseteq C$, $C \sqsubseteq D$, $E \sqsubseteq C$
    - 4 tests gone!
  - We can look for cheap consequences
    - E.g., $A \sqsubseteq C \sqcap D$ immediately gives $A \sqsubseteq C$, $A \sqsubseteq D$
      - Must take care about $\top \sqsubseteq A, C, D$ or $\top \sqsubseteq A, C, D$
- Exploit internals
  - For any SAT test we can
    - extract a representation of a model
    - if we have such a "pseudo-model" of C and of $\neg D$
      - We can see if they merge to form a new model
        - » Done!