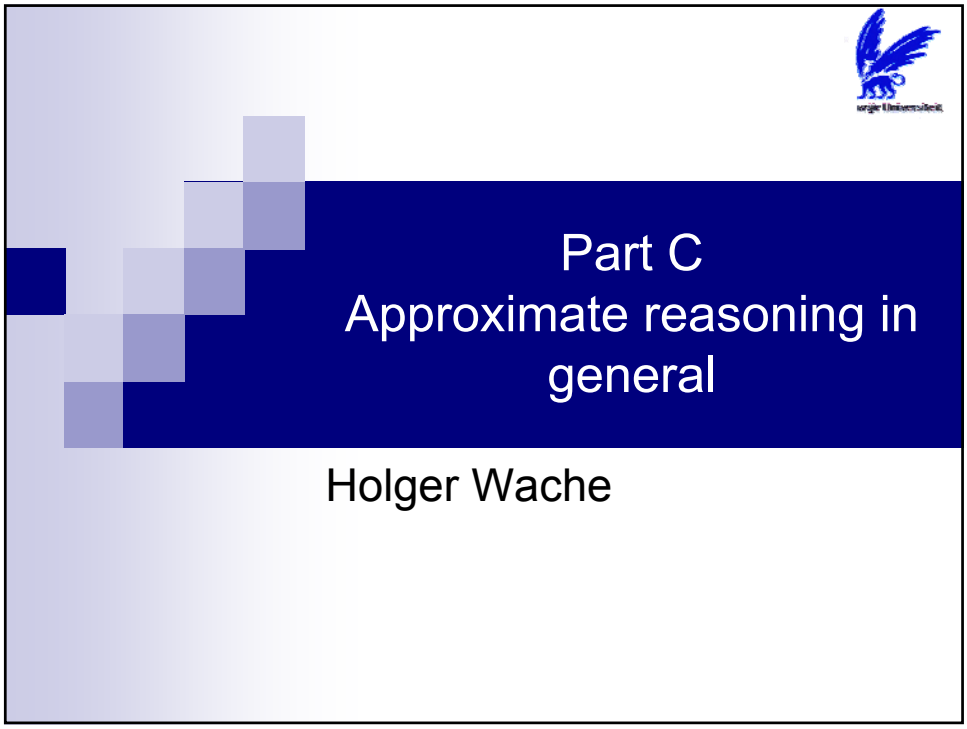
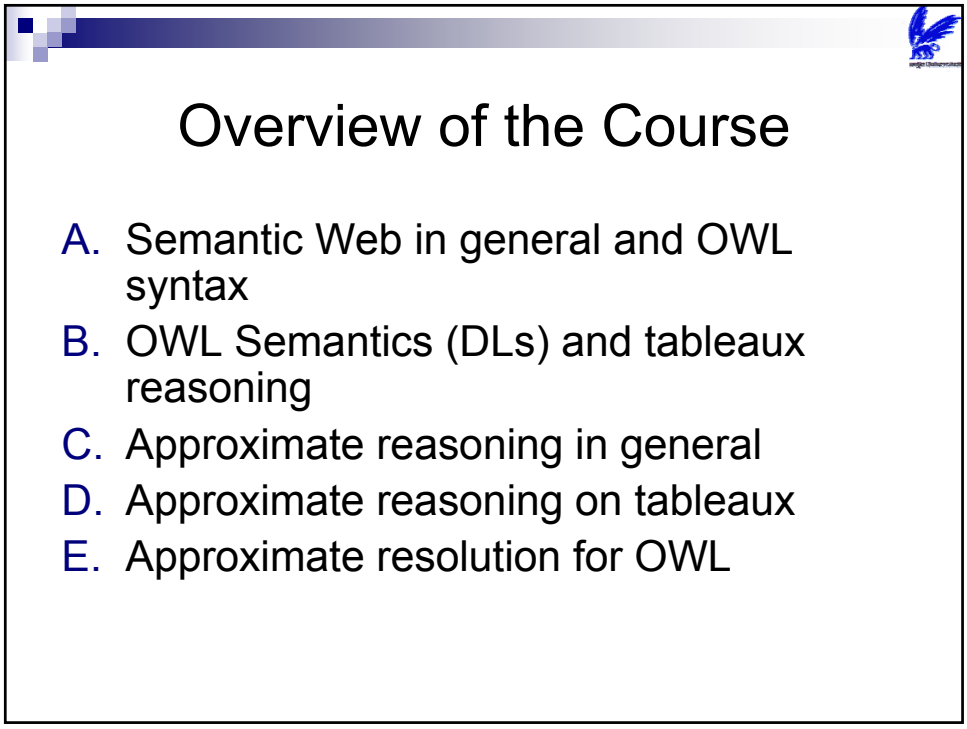
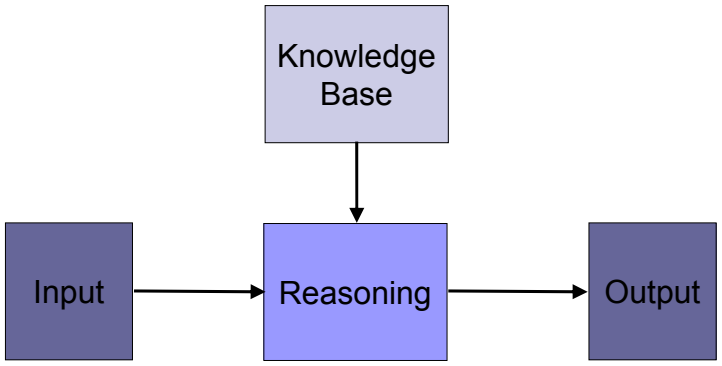

Part C
Approximate reasoning in general
Holger Wache


Overview of the Course
A. Semantic Web in general and OWL syntax
B. OWL Semantics (DLs) and tableaux reasoning
C. Approximate reasoning in general
D. Approximate reasoning on tableaux
E. Approximate resolution for OWL

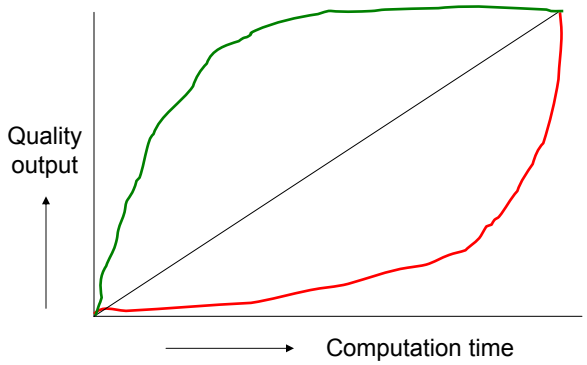
Knowledge-based Systems



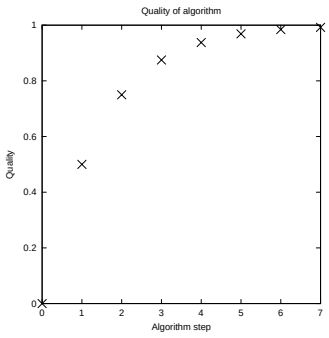
Motivation behind Approximation

- Reducing complexity
  - Reasoning under time pressure
  - Reasoning with other limited resources
- Reduce/increase number of solutions
- Reasoning that is not “perfect” but “good enough”

Anytime Reasoning



Reasoning Methods: Anytime Algorithms

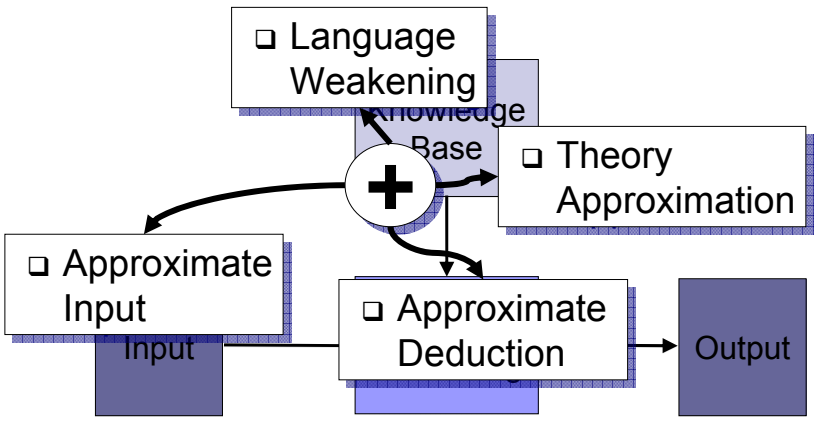


- **Measurable Quality** of the approximate result
- **Recognizable Quality** can be determined at run time
- **Monotonicity** over time and input quality
- **Consistency**
- **Diminishing returns** with more improvements in the beginning
- **Interuptibility** at any time
- **Preemptability** ensures algorithm can be suspended and resumed

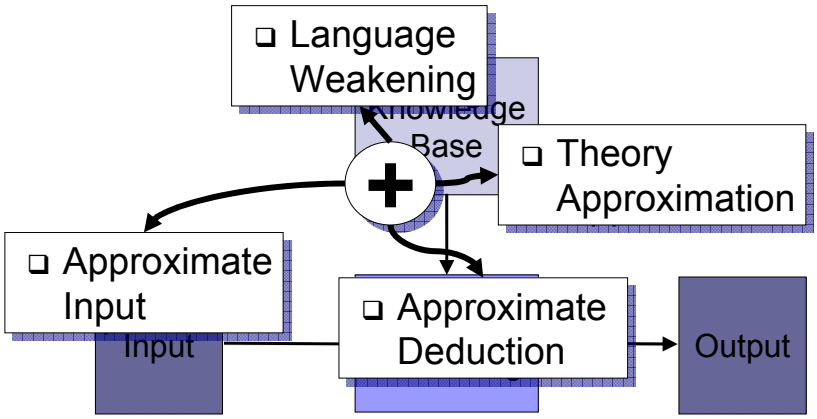
Types of Approximation

- Numerical
- Logical
  - Soundness
  - Completeness

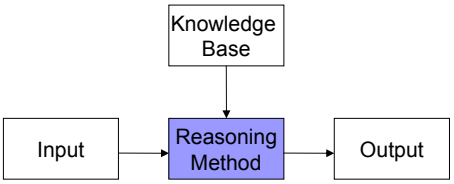
Approximation Approaches



Approximation Approaches



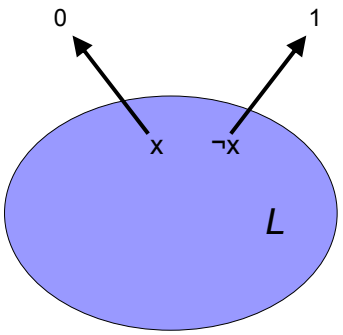
Guidelines for Reasoning Methods



- **Semantically well-founded** providing a clear answer to the problem
- **Computationally attractive** resulting in an easier computation of approximate answers
- **Improvable** approximate answers
- **Dual** sound and complete
- **Flexible** to apply to different problems

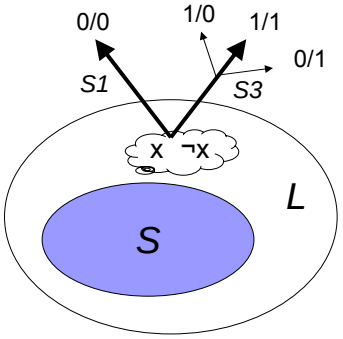
Logical Interpretation and Entailment

- *Interpretation*: interpret predicate as true or false
- *Entailment*: everything which is interpreted as true



Approximate Interpretation by Cadoli-Schaerf

- *S-1*-entailment: interpret everything outside of *S* as *false*
- *S-3*-entailment: interpret everything outside of *S* as *true* (or normal)



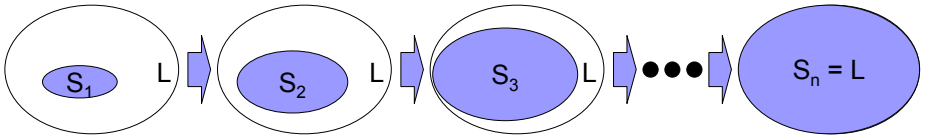
Approximate Entailment by Cadoli-Schaerf

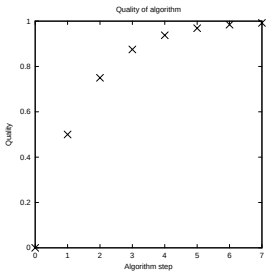
$\models_s^1$ $\models$ $\models_s^3$

Sound
But incomplete

Complete
But unsound

S1/S3-Entailment & Anytime





$\models_0^\top, \models_1^\top, \models_2^\top, \ldots \models_{n-1}^\top, \models$

- Anytime behavior when S_i will be enlarged continuously
- Interesting Feature:
  Reusing proof from previous level

Approximate Entailment

- Semantically well-founded
- Computationally attractive
- Improvable
- Dual
- Flexible

Approximate Entailment

- Unclear effect
- Parameter *S* is crucial for approximate behaviour
- Almost no quantitative analysis

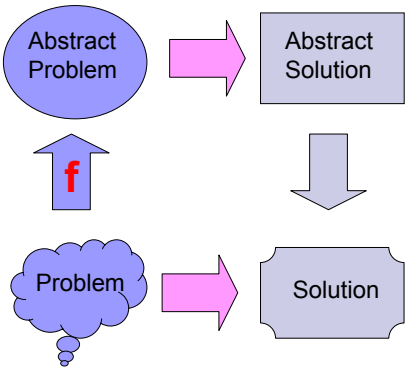
Boolean Constraint Propagation (BCP)

- Variant of unit resolution
  - Resolution only with at least one literal (P or ¬ P)
- Example:
  Γ = {¬ P, P ∨ ¬ Q, P ∨ Q ∨ ¬ R, P ∨ Q ∨ R}
  - ¬ Q from ¬ P, P ∨ ¬ Q
  - ¬ R from ¬ Q, ¬ P, P ∨ Q ∨ ¬ R
  - {} from ¬ P, ¬ Q, ¬ R, P ∨ Q ∨ R
- $\Gamma \vDash_{BCP} \phi$ iff $\{\} \in \Gamma \cup \{\neg \phi\}$

Variants of BCP

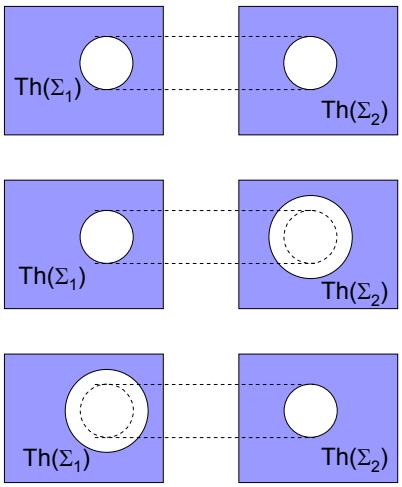
- Clausal BCP (restricted to clauses)
  sound but incomplete
  - CNF-BCP
  - Prime-BCP (intractable)
- Formula BCP (intractable)
- Fact Propagation

Approximation through Abstraction



- Process of mapping a representation of a problem onto a new representation
- Helps deal with the problem in the original search space by perserving certain desirable properties
- Is simpler to handle
- Abstraction: $\alpha \rightarrow f(\alpha)$

Forms of Abstractions



- TC(onstant)-Abstraction
  $\alpha \in Th(\Sigma_1)$ iff
  $f(\alpha) \in Th(\Sigma_2)$
- TI(ncrease)-Abstraction
  if $f(\alpha) \in Th(\Sigma_2)$
  then $\alpha \in Th(\Sigma_1)$
- TD(ecrease)-Abstraction
  if $\alpha \in Th(\Sigma_1)$
  then $f(\alpha) \in Th(\Sigma_2)$

Example: Predicate Abstraction

- Base Theory:
  - JapaneseCar(X) => Car(X)
  - EuropeanCar(X) => Car(X)
  - Toyota(X) => JapaneseCar(X)
  - BMW(X) => EuropeanCar(X)
- Extension
  - EuropeanCar(X) => Fast(X)
  - JapaneseCar(X) => Reliable(X)

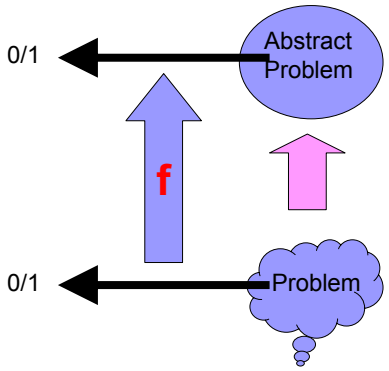
- Abstraction
  - *ForeignCar*(X) => Car(X)
  - Toyota(X) => *ForeignCar*(X)
  - BMW(X) => *ForeignCar*(X)
- Extension
  - *ForeignCar*(X) => Fast(X)
  - *ForeignCar*(X) => Reliable(X)

Problem with Syntactic Abstraction

- Abstraction includes:
  - BMW(X) => Reliable(X)
  - Toyota(X) => Fast(X)
- Stronger
  - ForeignCar(X) => (Fast(X) ∨ Reliable(X))

- Captures the final result of abstraction
- BUT:
  does not capture the underlying justification that leads to the abstraction
- Which is the best (TD)-Abstraction?

Semantic Abstraction



- Interpretations I_1, I_2
  - $I_1 \models Th(\Sigma_1)$
  - $I_2 \models Th(\Sigma_2)$
- Semantic Abstraction
  $f(I_1) = I_2$
- Model Increasing Abstraction
  if I_1 is model of $Th(\Sigma_1)$ then $f(I_2)$ is model of $Th(\Sigma_2)$

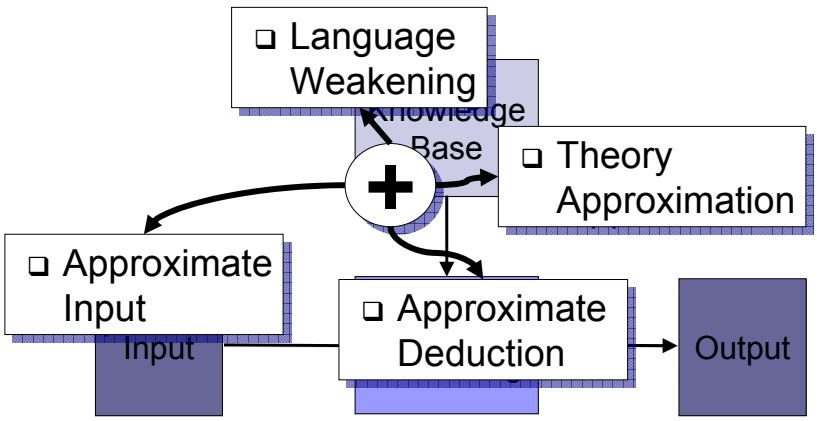
Example (cont.)

- Predicate Abstraction:
  - I(ForeignCar) = I(EuropeanCar) ∪ I(JapaneseCar)

Best Semantic Abstraction

- Strongest MI Abstraction
  $Th(\Sigma_2) = \{\sigma \mid \text{all models } I_1 \text{ of } Th(\Sigma_1): f(I_1) \models \sigma\}$
- Central question: How to find syntactic formulas for best Semantic Abstraction
  - ForeignCar(X) = EuropeanCar(X) ∪ JapaneseCar(X)
- Obvious in Description Logics?

Approximation Approaches



Approximate Input

- Approximating Terminological Queries
- Top-k querying

Conjunctive Queries

$Q \leftarrow q_1 \wedge \ldots \wedge q_n$ With q_i: x:C or (x,y) : R

C = concept name
R = role name
x,y = variables or constants

- Developed for description logics
- Most expressive query language

Example

$$
\begin{aligned}
Human &\sqsubseteq \top \\
Male &\sqsubseteq Human \\
Female &\sqsubseteq Human \\
Organization &\sqsubseteq \top \\
University &\sqsubseteq Organization \\
Title &\sqsubseteq \top \\
father\text{-}of &\sqsubseteq Male \times Human \\
works\text{-}at &\sqsubseteq Human \times Organization \\
degree &\sqsubseteq Human \times Title \\
granted\text{-}by &\sqsubseteq Title \times University \\
vu &: University
\end{aligned}
$$

$$
\begin{aligned}
Q(X) \leftarrow\ & (X,Y) : father\text{-}of \wedge (Y,Z) : works\text{-}at \wedge Y : Female \wedge \\
& (Y,V) : degree \wedge (V, vu) : awarded\text{-}by
\end{aligned}
$$

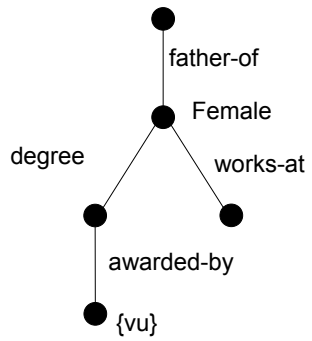
Approximating the Query

- Hypothesis: Less complex queries can be answered in shorter time.
- Query containment $Q^i \sqsupseteq Q^j$:
  - results(Q^j) $\subseteq$ results(Q^i)
- Create a sequence of queries $Q^1, \ldots, Q^m$ with
  - $i < j \Rightarrow Q^i \sqsupseteq Q^j$
  - $Q^m = Q$

How to determine the query sequence

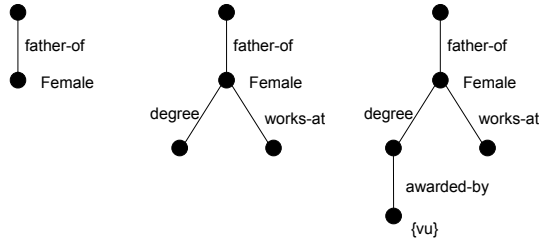
- Observation:
  Q^i contains less conjuncts than Q^j
  $\Rightarrow Q^i \sqsupseteq Q^j$
- Q^0 = empty
- $Q^{i+1} = Q^i \wedge q_k \wedge \ldots \wedge q_l$
- How to determine which conjuncts $q_k \wedge \ldots \wedge q_l$ are included in Q^{i+1}

Query Graph (Example)



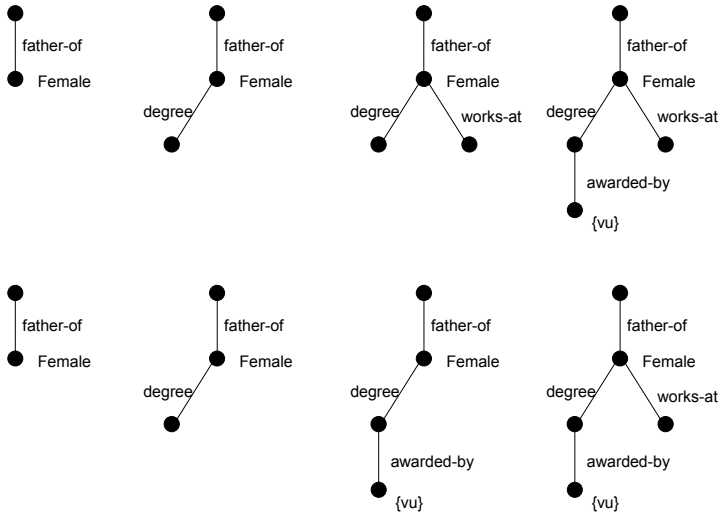
$$Q(X) \leftarrow (X,Y) : father\text{-}of \wedge (Y,Z) : works\text{-}at \wedge Y : Female \wedge (Y,V) : degree \wedge (V,vu) : awarded\text{-}by$$

Strategy I: Node expansion



$$Q^1(X) \leftarrow (X,Y) : father\text{-}of \wedge Y : Female$$
$$Q^2(X) \leftarrow Q^1(X) \wedge (Y,Z) : works\text{-}at \wedge (Y,V) : degree$$
$$Q^3(X) \leftarrow Q^2(X)(V,vu) : awarded\text{-}by$$

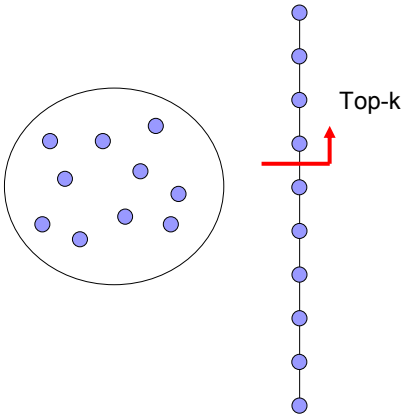
Strategy II: Search



Approximate Input

- Approximating Terminological Queries
- Top-k querying

Top-k queries



- Well known in databases
- Extends traditional database retrieval
- Instead of returning an unordered set of results also rank the results
- Top-K means return only the k best results
- Query language SQL extended to facilitate rank and/or scoring algorithm

Ranking

- Normally a (normalized) function

$$F \rightarrow [0..1]$$

- Sources for ranking
  - Google's Pagerank
  - User-preferences, e.g. user specify in which query predicates he is more interested in

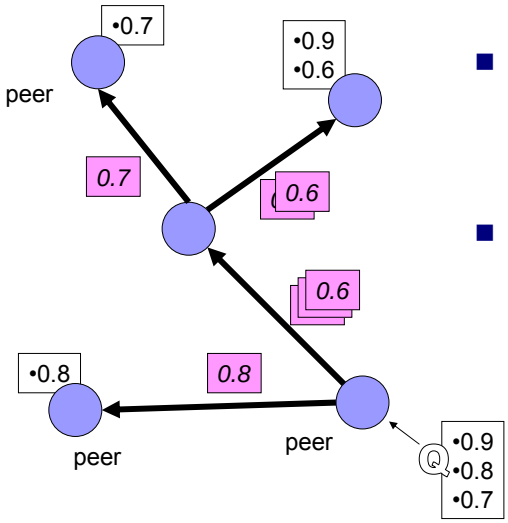
Naïve Algorithm for Top-k

- Naïve algorithm
  1. Retrieval all answers
  2. Order them according ranking function
  3. Return the best k results
- Problem
  - Too much unnecessary data accesses

Optimal Solutions

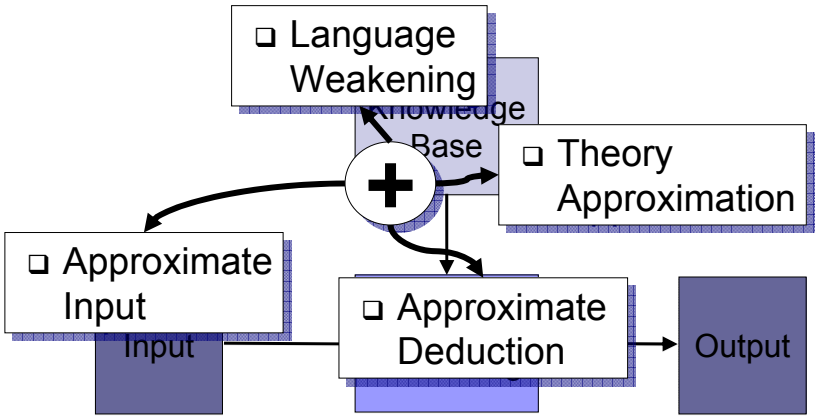
- Best solution: only k data accesses
- With respect to the necessary data access are known for
  - Multimedia retrieval
  - Databases, and
  - Web searches

Top-k in a peer-to-peer setting



- More Web-alike: assuming many sources for storing relevant answers
- Challenge: minimizing data transfer between peers
  - Each peer returns top-k answers

Approximation Approaches



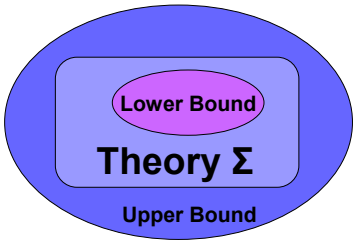
Knowledge Compilation

- deals with translating the knowledgebase such that the computational complexity of reasoning decreases.
- goal of knowledge compilation is to translate the knowledge into (or approximate it by) another knowledge base with better computational properties.

Approaches for Knowledge Compilation

- *Language Weakening*
  restrict the language used to represent knowledge
- *Theory Approximation*
  compile the theory into another "easier" theory (although still expressed in the same language).

General Principle of Approximate Knowledge Compilation



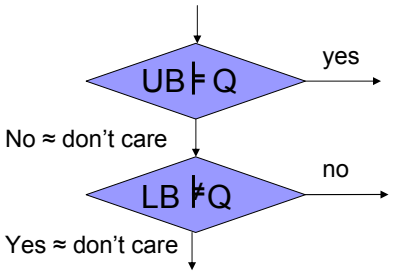
- Compute lower and upper bound

  $M(\Sigma_{lb}) \subseteq M(\Sigma) \subseteq M(\Sigma_{ub})$

  with M is set of models

- Good approximations
  - Greatest Lower Bound (GLB)
  - Least Upper Bound (LUB)

Using upper and lower bounds



- GLB is complete (but unsound)
  LUB is sound (but incomplete)
- Approximation:
  - Upper bound: if UB ⊨ Q then Σ ⊨ Q
  - Lower bound: if LB ⊭ Q then Σ ⊭ Q

Using Horn Approximations

- Translating *any* theory Σ into a Horn theory
  - Theory Σ must be in clause form
- Approximation when no exact translation exists
- Compute the unique LUB and one GLB (of many existing)

Horn-strengthening

- A Horn-Clause C_H is a Horn-strengthening of a clause C iff
  - $C_H \subseteq C$ and
  - There is not C'_H such that $C_H \subset C`_H \subseteq C$
- Examples: {p, ¬r} and {q, ¬r} are Horn-strengthening of {p, q, ¬r}

Computing GLB

GLB Algorithm
Input: a set of clauses $\Sigma = \{C_1, C_2, \ldots, C_n\}$.
Output: a greatest Horn lower-bound of Σ.
begin
  L := the lexicographically first Horn–strengthening of Σ
  loop
    L' := lexicographically next Horn–strengthening of Σ
    if none exists **then exit**
    if $L \models L'$ **then** $L := L'$
  end loop
  remove subsumed clauses from L
  return L
end

Lexicographical ordering

- Let C_i^j be the j-th Horn-strengthening of a clause C_i and $\Sigma = \{C_1, C_2, \ldots, C_n\}$
  1. $\{C_1^1, C_2^1, \ldots, C_n^1\}$
  2. $\{C_1^2, C_2^1, \ldots, C_n^1\}$
  3. $\{C_1^1, C_2^2, \ldots, C_n^1\}$
  4. $\{C_1^2, C_2^2, \ldots, C_n^1\}$
  5. …

Example

$\Sigma = (a \vee b) \wedge (\neg a \vee b \vee c)$

1. $L = (a) \wedge (\neg a \vee b) \equiv a \wedge b$
   $L` = (b) \wedge (\neg a \vee b) \equiv b$
2. $L \models L' \Rightarrow L := L'$
3. $L' = (a) \wedge (\neg a \vee c)$
4. No entailment $\Rightarrow$ no change
5. $L' = (b) \wedge (\neg a \vee c)$
6. No entailment $\Rightarrow$ no change
7. Remove redundant clause $(\neg a \vee b)$ from $L = (b) \wedge (\neg a \vee b) \equiv b$
8. Return $L = b$

GLB algorithm is anytime

- Computation of GLB can interrupted directly after initialization of L.
- L is improved during loop
- Interrupting will return the best approximation of GLB.

Prime implicates

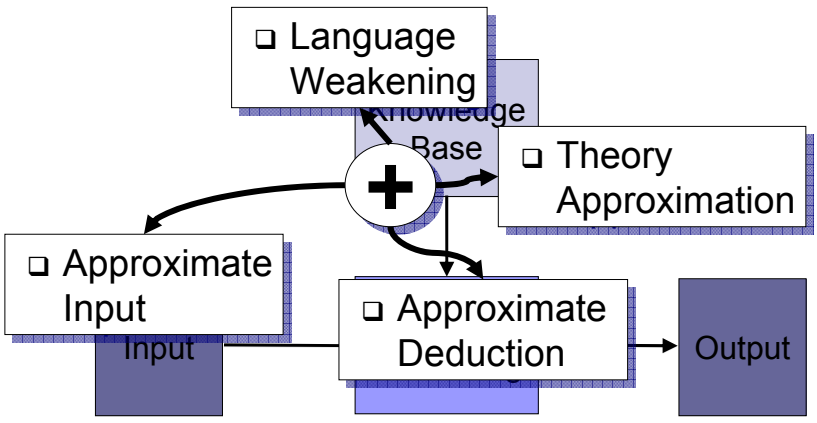
- Prime implicate C
  1. $\Sigma \vdash C$
  2. not $\exists$ C': $\Sigma \vdash$ C' and C' $\subset$ C
- Set of Prime implicates *PI*(Σ)
- Reasoning simplified

$$\Sigma \vdash C \quad \Leftrightarrow \quad C' \in PI(\Sigma): C' \subseteq C$$

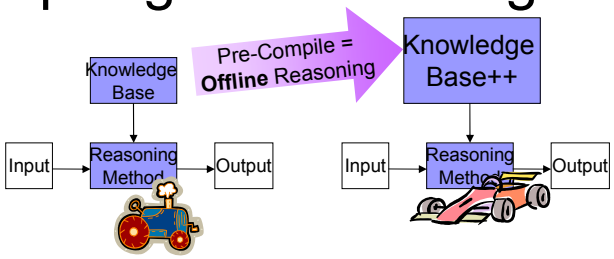
LUB and Prime implicates

- The LUB of Σ is logically equivalent to the set of all Horn prime implicates of Σ
- Using resolution to compute LUB
- CAUTION: LUB can be of exponential size (and is very time consuming!)

Approximation Approaches



Compiling the knowledgebase



- Exact Knowledge Compilation
- Approximate Knowledge Compilation

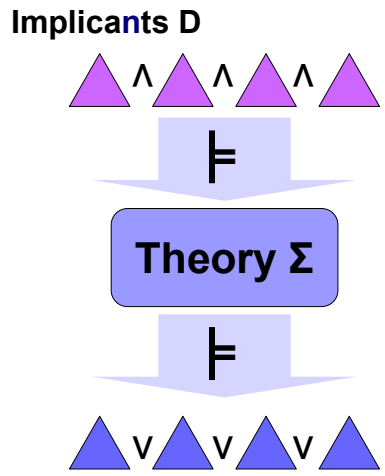
WARNING:
Literature: Knowledge Compilation =
here: Theory Approximation

Obvious Knowledge Compilation: Prime implicates

- Prime implicate C
  1. $\Sigma \vdash C$
  2. not $\exists$ C': $\Sigma \vdash$ C' and C' $\subset$ C
- Set of Prime implicates $PI(\Sigma)$
- Reasoning simplified

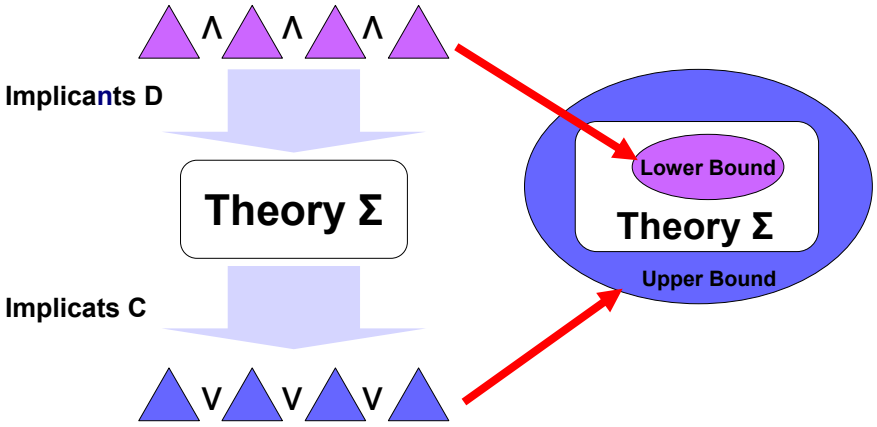
$$\Sigma \vdash C \quad \Leftrightarrow \quad C' \in PI(\Sigma): C' \subseteq C$$

Exact Knowledge Compilation



- Prime implicants $D \models \Sigma$ and
  Prime implicates $\Sigma \models C$
  Con-/Disjunction of Literals
- How to compute
  - Directly
  - Derivable by unit resolution
  - W.r.t. a tractable theory

Anytime Variants of Exact Methods



Thank you for your attention!