Goal: Computationally Efficient Knowledge Representation Systems

Knowledgerepresentationsystem:
Knowledgebasecontainsfactsabouttheworld.

Inferencemechanismintersnewfactsfromstoredones.

"Commonsenseknowledge"
Knowledgebasecontainsfactsabouttheworld.

Make implicit knowledge explicit.

Modular design for intelligent systems.
Modulesforperception,action,etc.

Querytheknowledgerepresentationsystemasneeded.

Goal: Computationally Efficient Knowledge Representation Systems

Knowledgerepresentationsystem:
Knowledgebasecontainsfactsabouttheworld.
Representing Knowledge

Logic: propositional, first-order, terminological, ...

Well understood semantics.

Declarative.

Central problem: Inference is intractable.

Used for commonsense theories.

Qualitative process theory (Forbus 1984)

Ontology of liquids (Hayes 1985)
Expressiveness versus Complexity

Direct tradeoff between expressiveness and tractability of a representation language.

(Levesque and Brachman 1985)

Standard databases expressively inadequate for disjunctive and/or incomplete information.

Problem:

Relational databases: restricted but efficient.

First-order logic: expressive but intractable.

Compare

(Levesque and Brachman 1985)

Expressiveness versus Complexity
Dealing with Complexity

1. Restricting the Language

Dealing with complexity

Disadvantage: Inference mechanisms often too weak.
Example: Four-valued semantics (no modulus ponens).
(Levesque 1984; Frisch 1986)

2. Incomplete reasoning: non-standard semantics

Example: Restricted terminological logics.
(Levesque and Brachman 1985; Nebel et al. 1990)

Disadvantage: Sublanguage often not sufficiently expressive.

Disadvantage: Sublanguage often not sufficiently expressive.

Example: Restricted terminological logics.
5. Knowledge Compilation

Alternative approach:

Disadvantage: unsound.

Information. (Etherington et al., 1989; Selman, 1990)

Example: use defaults to remove disjunctive

(Levesque, 1986)

4. Vivid Reasoning

Disadvantage: unclear what can / cannot be inferred.

Example: run theorem prover for limited amount

(Doyle and Patil, 1991)

3. Incomplete Reasoning: Resource bounded
Knowledge Compilation

Translate knowledge given in some general representation into a tractable, restricted language.

Exact translation often not possible.

Yet retain soundness & completeness.

Can approximate original theory.

In answering queries.

Translate knowledge given in some general representation

Source language $\rightarrow$ Target language

Language into a tractable, restricted language.

Knowledge Compilation
Outline

Propositional case
Defining Horn approximations
Propositional case

Other tractable target languages

Extensions

Algorithms and complexity
Properties

Terminological logics
Propositional Theories

Source: Clausal Proositional Theories

Inference: NP-Complete

Target: Horn Theories

Model (of a theory): a truth assignment (under which the theory evaluates to "true")

Notation

Clause: disjunction of literals.
Clausal theory: conjunction of clauses (CNF).
Horn clause: at most one positive literal.
Negative clause: \((-q \lor \neg a)\)
Equivalence: \((q \lor a) \equiv c\)
Example: \((-q \lor \neg a) \land c\)

Notation

Inference: Linear Time

Target: Horn Theories.

Source: Clausal Propositional Theories.
Horn Approximations: Model Theory

\[ \models_{\text{ub}} \models_{\text{lb}} = \text{Upper-bound Horn approximation.} \]

\[ \models_{\text{lb}} = \text{Lower-bound Horn approximation.} \]

\[ \models = \text{Original CNF theory.} \]
Definition: Horn Bounds

Where is a set of clauses and $M/satisfying truth assignments/
is the set of models of $satisfying truth assignments/$.

Define $lb$ is a Horn lower bound of $ub$ is a Horn upper bound of $lb$ and $ub$ are sets of Horn clauses and $M/lb$ and $M/ub$ are sets of models of $lb$ is a Horn upper bound of $ub$ is a Horn lower bound of

Equivalently

\[ q^n \models \exists \models q^l \exists \models \]

If $lb$ and $ub$ are sets of Horn clauses and $lb$ is a Horn upper bound of $ub$ is a Horn lower bound of

Definition: Horn Bounds

Where $\exists$ is a set of clauses satisfying truth assignments (and $M/\exists$ is the set of models of $\exists$)
\( \text{glb} \) and \( \text{lub} \) are Horn approximations of \( \exists \).

\( \text{Is unique for } \exists. \)

Equivalently: Strongest Horn theory implied by \( \exists \).

No set \( \exists \) of Horn clauses such that

\( \text{glb} \) \( \subseteq \) M(\( \exists \)) \( \subseteq \) M(\( \exists \))

Equivalently: A weakest Horn theory that implies \( \exists \).

No set \( \exists \) of Horn clauses such that

\( \text{glb} \) \( \subseteq \) M(\( \exists \)) \( \subseteq \) M(\( \exists \))

\( \text{Not unique for } \exists. \)

Equivalently: A weakest Horn theory that implies \( \exists \).

No set \( \exists \) of Horn clauses such that

\( \text{glb} \) \( \subseteq \) M(\( \exists \)) \( \subseteq \) M(\( \exists \))

\( \text{glb} \) is a greatest Horn lower-bounded (GLB)
\[
\begin{align*}
\text{LUB:} & \quad & \text{GLB:} \\
(a \land q \land \neg a \land \neg c) \lor (a \land \neg a) & = & c & = & \exists \\
\text{Horn upper-bound:} & \quad & \text{GLB:} \\
(c \lor q \land \neg a) & = & a & \lor q & \land c & = & c \\
\text{Horn lower-bound:} & \quad & \text{GLB:} \\
(c \lor q) & = & a & \lor q & \land c & = & \exists \\
\end{align*}
\]

Example
Using Approximations for Query Answering

to a series of queries.

Improvement in overall response time

Queries answered in linear time lead to

(Or return "don’t know.")

• Otherwise, use \( \neq \) directly.

(Linear time.)

• If \( \not\subseteq \) \( \not\subseteq \) then \( \not\subseteq \).

(Linear time.)

• If \( \not\sqsubseteq \) \( \subseteq \) then \( \subseteq \).

\[ \neq \]
Answer in linear time.

Query can be arbitrary CNF formula.

Horn Approximations:

target language.

Query languages can be more expressive than query languages.
Generalizes (Borgida and Etherington 1989)

\[ \text{LUB} \text{ is abstraction of facts + original background knowledge.} \]

Consider background knowledge:

**Fact:**

\[
\text{doctor}(x) \subseteq \text{professional}(x)
\]

\[
\text{lawyer}(x) \subseteq \text{professional}(x)
\]

LUB can be viewed as an abstraction.

Properties of Horn Approximations
GLB can be viewed as a specialization

As a counterexample.

(Levesque 1986), mental models (Johnson-Laird 1980).

As a positive example.

Compare geometry theorem proving (Gelernter 1969).

Not sound when used in this way.

Jump to conclusion that doctor(Jill).

GLB is a set of models, so generalizes vivid reasoning.

GLB can be viewed as a specialization.
Computing Horn Approximations

**Theorem:** Let $\mathcal{C}$ be a set of clauses. The GLB of $\mathcal{C}$ is consistent if $\mathcal{C}$ is consistent. (Similarly for LUB.)

**Theorem:** Let $\mathcal{C}$ be a set of clauses. The GLB of $\mathcal{C}$ is consistent if $\mathcal{C}$ is consistent.
Algorithm: search space of Horn-strengthenings.

Horn-strengthening of $\Xi$.

Theorem: Each GLB of $\Xi$ is equivalent to some among others:

$$s \lor (\mu \land d)$$

has strengthening $$(q \land s) \lor (\mu \land b \land d)$$

Horn-strengthening of a theory:

$$\mu \land b$$

and $$\mu \land d$$

has Horn-strengthenings $\mu \land b \land d$

Horn-strengthening:

Computing the GLB
Lemma

Where $H$ is a Horn theory, and $C$ is any clause:

If $H \models C$, then $H \models \text{Horn-strengthening of } C$.

Proof

Because the resolvent of Horn clauses is Horn, $C'$ is Horn.

Therefore there is some $C_H$ that is a Horn-strengthening of $C$.

So there is some clause $C_0$ that follows from $H$ by resolution, such that $C_0 \supset C'$.

By completeness of resolution, there is some clause $C_H'$ such that $C_H' \supset C'$.

Therefore $H \models C_H'$.

Where $H$ is a Horn theory, and $C$ is any clause:
Therefore, $\exists \text{glb} \equiv \exists \cdot$

But $\exists \text{glb}$ is a greatest (weakest) Horn lower-bound.

$\exists \text{glb} \models \exists \cdot$ $\exists \text{glb}$ entails some Horn-strengthening $\exists$, of $\exists$

$\exists \text{glb}$ is Horn, so by lemma

$\text{GLB} = \text{weakest Horn-strengthening}$
procedure GLB(\(Z\))

begin

end

end

remove subsumed clauses from \(T\)

end loop

begin loop

\(I =: I\) if \(I = I\) then \(I = I\) if none exists then exit.

Horn-strengthening of \(Z\)

\(I =: I\) := lexicographically next

begin loop

Horn-strengthening of \(Z\)

\(I =: I\) := the lexicographically first

begin

/* Computes some Horn-greatest lower-bound of \(Z\) */

procedure GLB(\(Z\))

begin

end

end

end loop

begin

end

end loop

end

end

end

end

end

end
Example of GLB Algorithm

Because

\[(p \land a^-) \lor (c \land a^-) = \epsilon_T\]
\[(c \land a^-) \lor (c \land a^-) = \epsilon_T\]
\[(q \land a^-) \lor (c \land a^-) = \epsilon_T\]

Horn-strengthenings:

\[(p \land c \land q \land a^-) \lor (c \land a^-) = \exists\]

algorithm returns

\[\exists_T \neq \epsilon_T\]
\[\epsilon_T \equiv T\]
Properties of the GLB Algorithm

- Anytime. Algorithm may be stopped at any time.
- Length of GLB \( \leq \) length of original theory.
- Lower-bound improves over time.
- Find a lower-bound (not necessarily a GLB).
Computing the LUB

Basic Strategy:
Compute all resolvents of original theory, and collect all Horn resolvents.

Problem:
Even a Horn theory can have exponentially many Horn resolvents.

Solution:
Resolve only pairs of clauses containing at least one non-Horn clause.

Method is complete. (Selman and Kautz 1991)
Algorithm does not resolve (1) + (2)

\[ \varphi \lor q \equiv q \lor (\varphi \land q \neg) \lor (q \land p \neg) \]

Answer is

\[ \varphi \land p = (3) + (2) \]
\[ q = (3) + (1) \]

Resolvents:

\[ (q \land p) \lor (\varphi \land q \neg) \lor (q \land p \neg) = \Xi \]

Example of Computing LUB
Properties of the LUB algorithm

- New letters can sometimes reduce size.
- Exponentially larger LUB.
- Can construct non-Horn theories with exponentially larger LUB.
- No space blow-up for Horn.
- Anytime.

27
No smaller equivalent set of clauses exist:

Size LUB (2^n)

\( \text{ReadsMcCarthy} \lor \text{ReadsDennett} \lor \text{ReadsKosslyn} \lor \text{CogSci} \)

\( \text{CompSci} \lor \text{Phil} \lor \text{Psych} \lor \text{CogSci} \)

\( \text{LUB is} \)

\( \text{ReadsKosslyn} \lor (\text{Psych} \lor \text{CogSci}) \)

\( \text{ReadsDennett} \lor (\text{Phil} \lor \text{CogSci}) \)

\( \text{ReadsMcCarthy} \lor (\text{CompSci} \lor \text{CogSci}) \)

\( \exists \) is an example
Size \( \text{LUB} \( \omega \) \).

\[
\begin{align*}
\text{ReadMcCarthy} \subset \text{PsychBuff} \\
\text{Psych} \subset \text{PsychBuff} \\
\text{PhilBuff} \subset \text{PhilBuff} \\
\text{PsychBuff} \subset \text{PhilBuff} \\
\text{PhilBuff} \subset \text{PhilBuff} \\
\text{CompSci} \subset \text{CompSciBuff} \\
\text{CompSciBuff} \subset \text{CompSciBuff} \\
\text{CompSciBuff} \subset \text{CompSciBuff} \\
\text{CompSciBuff} \subset \text{CompSciBuff}
\end{align*}
\]

LUB becomes

\[
\text{CompSciBuff} \equiv (\text{Psych} \land \text{ReadMcCarthy})
\]

\[
\text{PhilBuff} \equiv (\text{PhilBuff} \land \text{ReadMcCarthy})
\]

\[
\text{CompSciBuff} \equiv (\text{CompSci} \land \text{ReadMcCarthy})
\]

Shrinking LUB: Introduce new concepts
of the LUB always exist?

QUESTION: does a small (perhaps non-obvious) representation

Forming concepts for fast inference:

a tractable approximation of the original theory:

• So, new concepts are useful generalizations for obtaining

  E.g., computeBreedsMcCarthy

  have in common.

• New concepts capture what certain pairs of propositions

  in Lang(∅).

• New LUB and original LUB are equivalent on queries
What Do We Mean by "Size"?

Consider then any representation of the Horn LUB

- Schemata, etc.
- Structure sharing,

...is inherently large: there may be clever ways to encode a large set of clauses

Not sufficient for proving that something smallest equivalent set of clauses

Perhaps: "size" will mean size of

\[ x \equiv x, \text{ yet } x \text{ exponentially larger} \]

There are many equivalent clausal theories

What Do We Mean by "Size"?
The answer: There do exist theories whose Horn LUB is inherently large. Any representation of the LUB that enables polynomial time inference is exponentially larger than the theory.
Proof of Circuit Complexity

(For each m).

For any formula over m variables whose LUB can be used to solve 3-SAT can construct a single particular theory.

Proof of inherently intractable LUB's:

Can construct a single particular theory:

- same polynomial for all n.

(Equivalent: some polynomial algorithm)

circuit for inputs of length n

Non-uniform P: for every n, there is some polynomial

Proof: Circuit Complexity
Simultaneously, that not all the inputs set to "I" can hold simultaneously computes "I" if \( \text{LUB}(\mathcal{Z}) \) entails

\[ \exists \text{SAT over } m \text{ variables} \]

(2\(m\),3 inputs: specify any)

\[ \exists d \land \exists d \land \exists d \land \neg 1d \land \neg 2d \land \neg 3d \]

\[ \neg 1d \land \neg 2d \land \neg 3d \land \neg 1d \land \neg 2d \land \neg 3d \]

\[ \neg 1d \land \neg 2d \land \neg 3d \land \neg 1d \land \neg 2d \land \neg 3d \]

\[ \neg 1d \land \neg 2d \land \neg 3d \land \neg 1d \land \neg 2d \land \neg 3d \]
For the \( i \) (input) variables specified as a single clause made up

Any 3-CNF formula over \( m \) variables can be

where the \( \neg \) variables are new

\[
\{ \text{if } x, \text{ if } y \text{ are literals} \quad | \quad \neg z \land \neg y \land \neg x \}\lor = 3
\]

Define:

For 3-SAT formulas over a given set of \( m \) variables,

Universal 3-SAT Theory
Note: query is Horn, so LUB is complete!

\[
\text{LUB}(\exists t) = \exists (d_1 \land \exists d_2 \land \exists d_3 \land \exists d_4) \\

\text{iff} \\

\exists (d_1 \land d_2 \land d_3 \land d_4) \\

= \exists (d_1 \land d_2 \land d_3 \land d_4) \\

\text{iff} \\

\exists (d_1 \land d_2 \land d_3 \land d_4) \text{ is unsatisfiable} \\

Example
Still considered to be pretty unlikely:

Polynomial hierarchy collapses to \( \Sigma^p_2 \)

Weaker condition than \( \text{P=NP} \):

\( \text{NP} \subseteq \text{non-uniform P} \)

Existence of small LUB's for these universal theories

Relation to the Polynomial Hierarchy
Implications

GLB and LUB may be different kinds of theories for the particular input theory classes, pick most powerful class that is small. Could try to compile to several different tractable classes, that does guarantee it is small. May choose LUB from a different tractable

\( O(\max \text{ size}) \)

\( \text{Not Horn} \quad (b \land d) \quad \text{okay} \)

\( \text{2-SAT} \quad \text{conjunction 2 literal clauses} \)

\( O(\max \text{ size}) \)

\( \text{k-Horn} \quad \text{bound length of Horn clauses} \)

Implied
Extensions

- Other target languages:

  - 2-SAT (O(n^2) max size).
  - K-Horn (O(n) max size).

Similar to (Subramaniam and Genesereth 1987).

"Compiling away" parts of original theory.

"Irrelevant" propositions.

Clauses not containing given set of

First-order source and target languages:

Algorithms may not terminate.

GLB algorithm: Interleave comparison of Horn-

- Interleaving comparison of Horn-
Compile FL concepts to FL concepts.

FL: tractable (no role restrictions).

FL: intractable.

Terminological logics (Classic, Breachman et al.):

Other Formalisms: Terminological Logics
A query language can be more expressive than target language. Again, query language can be more expressive in polynomial time. Can determine subsumption between concept $\mathcal{F}_C$ and concept $\mathcal{F}_C$. Recent work by Lenzerini, et al. (1991) shows that...
Does Knowledge Compilation Really Work?

- Are costs always shifted to compilation time?
- Highly structured data.
- Planning problems
- Relatively unstructured data.
- Hard, randomly generated theories

Classes considered:

- Is there empirical evidence of savings?

- Or are such queries easy for original theory?
- Can the bounds answer „hard“ queries?

or can the compilation process itself be „paid off“?
Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!

Therefore: KC does not just "skim" easy queries!
Hard Random Theories

Test set: 40 random 3-CNF theories, ranging in size from 75 to 200 variables. Ratio of 4.3 clauses per variable yields weaker than the Horn LUB and GLB, but easier to compute and analyze.

We computed the unit clause LUB and GLB (Mitchell, Selman, and Levesque 1992) but not the best Horn bounds.

Note: unit clause bounds are also Horn bounds, but weaker than the Horn LUB and GLB, but easier to compute and analyze.

Hard Random Theories
Based on the size of the bounds obtained, we can exactly compute the percentage of all randomly generated queries that are answered by the bounds alone.

<table>
<thead>
<tr>
<th>Vars</th>
<th>Clauses</th>
<th>GLB size unit</th>
<th>LUB size unit</th>
<th>percent queries answered</th>
</tr>
</thead>
<tbody>
<tr>
<td>85%</td>
<td>83%</td>
<td>188</td>
<td>132</td>
<td>80%</td>
</tr>
<tr>
<td>66%</td>
<td>66%</td>
<td>62</td>
<td>62</td>
<td>150</td>
</tr>
<tr>
<td>79%</td>
<td>76%</td>
<td>93</td>
<td>57</td>
<td>100</td>
</tr>
<tr>
<td>88%</td>
<td>85%</td>
<td>71</td>
<td>53</td>
<td>322</td>
</tr>
</tbody>
</table>

Based on the size of the bounds obtained, we can exactly compute the percentage of all randomly generated queries that are answered by the bounds alone.
Execution Time.

<table>
<thead>
<tr>
<th>Vars</th>
<th>Clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>55</td>
<td>0</td>
</tr>
<tr>
<td>200</td>
<td>860</td>
</tr>
<tr>
<td>150</td>
<td>645</td>
</tr>
<tr>
<td>100</td>
<td>430</td>
</tr>
<tr>
<td>75</td>
<td>322</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tableau only</th>
<th>Tableau only</th>
<th>Bounds and Tableau</th>
<th>Tableau only</th>
</tr>
</thead>
<tbody>
<tr>
<td>binary</td>
<td>ternary</td>
<td>binary</td>
<td>binary</td>
</tr>
<tr>
<td>86</td>
<td>258</td>
<td>88</td>
<td>48</td>
</tr>
<tr>
<td>128</td>
<td>95</td>
<td>45</td>
<td>51</td>
</tr>
<tr>
<td>341</td>
<td>322</td>
<td>258</td>
<td>88</td>
</tr>
<tr>
<td>248</td>
<td>45</td>
<td>55</td>
<td>0</td>
</tr>
<tr>
<td>1084</td>
<td>61</td>
<td>860</td>
<td>200</td>
</tr>
<tr>
<td>86</td>
<td>51</td>
<td>645</td>
<td>150</td>
</tr>
<tr>
<td>128</td>
<td>45</td>
<td>430</td>
<td>100</td>
</tr>
<tr>
<td>341</td>
<td>55</td>
<td>322</td>
<td>75</td>
</tr>
<tr>
<td>248</td>
<td>0</td>
<td>860</td>
<td>200</td>
</tr>
<tr>
<td>1084</td>
<td>61</td>
<td>645</td>
<td>150</td>
</tr>
<tr>
<td>86</td>
<td>45</td>
<td>430</td>
<td>100</td>
</tr>
<tr>
<td>128</td>
<td>55</td>
<td>322</td>
<td>75</td>
</tr>
<tr>
<td>341</td>
<td>0</td>
<td>860</td>
<td>200</td>
</tr>
<tr>
<td>248</td>
<td>61</td>
<td>645</td>
<td>150</td>
</tr>
</tbody>
</table>

Implementation query algorithm, using the program, to handle queries on which bounds fail.

A version of the Davis-Putnam procedure, Tableau (Crawford and Auton 93).

Time in seconds to answer 1000 random queries (SGI Challenge).

To handle queries on which bounds fail,
Knowledge compilation might not be expected to work on unstructured, random formulas. However, even unit clause bounds gave great computational savings on average, over 100x faster on 3-literal queries.

On 200 variable theories, compilation time outperformed original goal of simply shifting execution time off-line.

After 420 3-literal queries — (approx. 1 hour) completely paid for compilation time — on average, compilation time over 100X faster on 3-literal queries.

Random Formulas: Summary
Planning is NP-complete (not just shortest-path).

Plans correspond to models of the theory.

(Kautz and Selman, 1992).

Encoded in the planning as satisfiability framework.

Making other nodes inaccessible (‘forbidden pairs’).

Moving to certain nodes consumes resources.

(Pollack 1989; Hendler 1990)

Domain: Robot moving in a graph-like environment.

Planning Problems
Example: In going from a to g in at most 5 steps, can the robot visit j?
Answer: No (by length of shortest path).
Example: In going from a to g in at most 5 steps, can the robot visit b?
Answer: Yes (a model contains path a-b-d-e-f-g).
Example: In going from \( a \) to \( g \) in at most 10 steps, must the robot visit \( e \)?

Answer: Yes (forbidden pairs eventually block all routes through area labeled “MAZE”).
Hand - 5 hand-constructed "non-obvious" queries.

"is the robot ever at (a specified) point?"
to predicates of the form

RandEver - 400 random binary queries, restricted
RandBin - 500 random binary queries.

Query test sets:

Compiling unit bounds takes 1.2 hours.

29,576 clauses.
576 variables.

Axioms for mapworld shown in previous figures use

Compiling SAT Encoding of Planning Domain
Planning Results

<table>
<thead>
<tr>
<th>Theory</th>
<th>Times in seconds, on an SGI Challenge.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theory</td>
<td>5</td>
</tr>
<tr>
<td>Theory</td>
<td>6</td>
</tr>
<tr>
<td>Theory</td>
<td>8</td>
</tr>
<tr>
<td>Theory</td>
<td>10</td>
</tr>
<tr>
<td>Theory</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HC</th>
<th>439</th>
</tr>
</thead>
<tbody>
<tr>
<td>3748</td>
<td></td>
</tr>
<tr>
<td>8953</td>
<td></td>
</tr>
<tr>
<td>1071</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time only</th>
<th>only time</th>
<th>bounds only time</th>
</tr>
</thead>
<tbody>
<tr>
<td>144</td>
<td>376</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>617</td>
<td>83</td>
<td>0</td>
</tr>
<tr>
<td>840</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3748</td>
<td>464</td>
<td>2013</td>
</tr>
<tr>
<td>8953</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>1071</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Of original theory and its LUB.

"TZ+LUB" time — using Tableau on conjunction

Theory has 576 variables, 29,576 clauses.

Times in seconds, on an SGI Challenge.

Number answered by bounds

Bounds only time

KC using TZ+LUB time

TZ+LUB time

KC Query time

Theory only time

Number of queries
Substitute sensing for theorem proving.

If willing to ignore queries not answered by the bounds, first test against bounds; if bounds fail, test against theory + LUB | 10X speedup.

Best performance: first test against bounds; if bounds fail, test against theory + LUB, and using a complete theorem prover. Similar benefit gained by simply conjointing theory and its LUB, and its LUB, and using a complete theorem prover.

Basic KC-Query algorithm increased speed by 2X to 4X.

Observations
Conclusions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

stronger Horn bounds worthwhile?  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Open question: is additional cost of computing approximations?  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Good performance obtained with unit clause approximations.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Significant speed-up occurs on both unstructured (random) and structured (planning) problems.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Success in shifting computational cost off-line: bounds answer many queries that would be hard to answer with any complete theorem prover.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We have begun to evaluate computational savings possible with knowledge compilation by theory approximation.

Conclusions
Summary and Conclusions

Proposed: "Knowledge compilation towards obtaining efficient knowledge representation systems.

Features:
- Generalizability (any extensional semantics)
- Efficiency improves over time
- Sound and complete inference
- "Model-based" reasoning
- Generalizes other work on abstraction and generalizes other work on abstraction and generalizes other work on abstraction and generalizes other work on abstraction and approximations based on two de
ing bounds.
- No restrictions on expressiveness of source language.

A proposal towards obtaining efficient knowledge compilation

Introductory knowledge compilation.