Goal: Computationally Efficient Knowledge Representation Systems

Knowledge representation system:

Knowledge base contains facts about the world.

"Commonsense knowledge"

Inference mechanism infers new facts from stored ones.

Make **implicit** knowledge **explicit**.

Modular design for intelligent systems.

Modules for perception, action, etc. query query the knowledge representation system as needed.

Representing Knowledge

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

- Declarative.
- Well understood semantics.
- Used for commonsense theories:
  - Ontology of liquids (Hayes 1985)
  - Qualitative process theory (Forbus 1984)

Central problem:

- Inference is **intractable**.

Expressiveness versus Complexity

Direct **tradeoff** between expressiveness and
tractability of a representation language.

(Levesque and Brachman 1985)

Compare

First-order logic: expressive but intractable.

Relational databases: restricted but efficient.

Problem:

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

$Wetness(h') \wedge Episode(h, h') \equiv$

$Merewet(h) \vee Drying(h) \vee Spreading(h)$

(Hayes 1985)

Dealing with Complexity

1. **Restricting the language**

   (Levesque and Brachman 1985; Nebel *et al.* 1990)

   Example: restricted terminological logics.

   Disadvantage: sublanguage often not sufficiently expressive.

2. **Incomplete reasoning: non-standard semantics**

   (Levesque 1984; Frisch 1986)

   Example: four-valued semantics (no modus ponens).

   Disadvantage: inference mechanism often too weak.

3. Incomplete reasoning: resource bounded

(Doyle and Patil 1991)

Example: run theorem prover for limited amount of time.

Disadvantage: unclear what can / cannot be inferred.

4. Vivid reasoning

(Levesque 1986)

Example: use **defaults** to remove disjunctive information. (Etherington *et al* 1989; Selman 1990)

Disadvantage: unsound.

Alternative approach:

5. Knowledge Compilation

Knowledge Compilation

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

source language $\longrightarrow$ **target language**

Exact translation often not possible.
Can **approximate** original theory
yet retain soundness & completeness
in answering queries.

Outline

- Propositional case
  - Defining Horn approximations
  - Properties
  - Algorithms and complexity
- Extensions
  - Other tractable target languages
  - Terminological logics

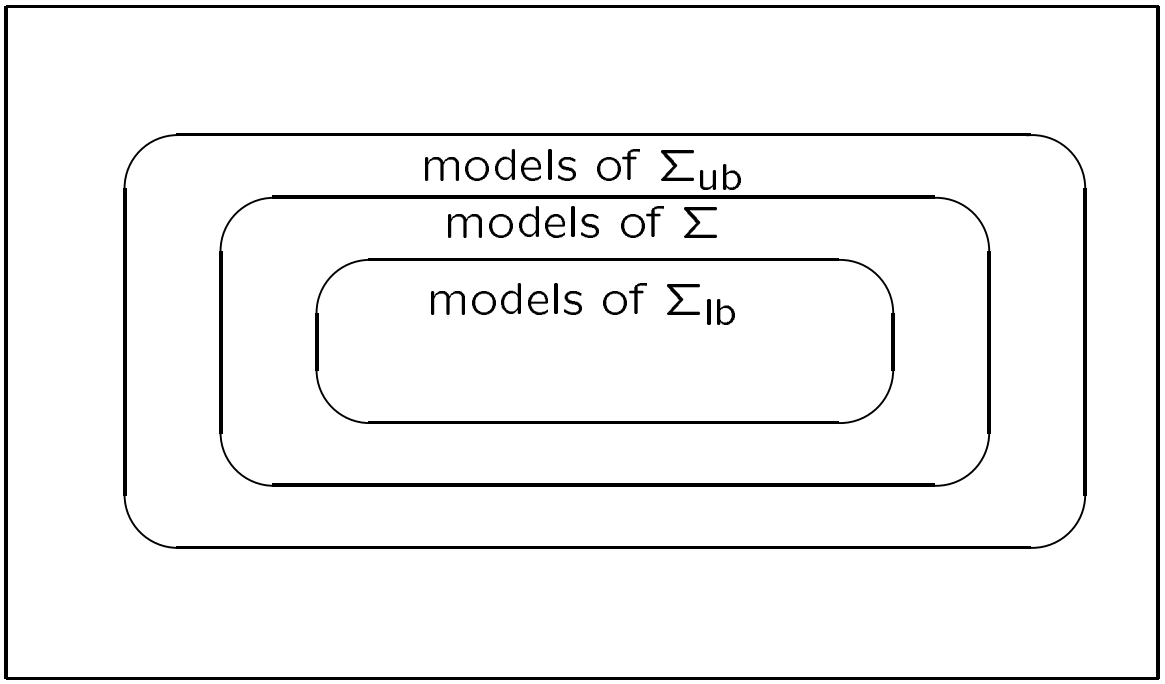
Propositional Theories

Source: clausal propositional theories.
Inference: NP-Complete.
Target: Horn theories.
Inference: linear time.

Notation

Clause: disjunction of literals.
Clausal theory: conjunction of clauses (CNF).
Horn clause: at most one positive literal.
Example: $(\neg a \vee \neg b \vee c)$
Equivalently: $(a \wedge b) \supset c$
Negative clause: $(\neg a \vee \neg b)$
Model (of a theory): a truth assignment (under which the theory evaluates to "true").

Horn Approximations: Model Theory



Σ = original CNF theory.

Σ_{lb} = Lower-bound Horn approximation.

Σ_{ub} = Upper-bound Horn approximation.

Definition: Horn Bounds

Where Σ is a set of clauses

and $\mathcal{M}(\Sigma)$ is the set of models of Σ

(satisfying truth assignments)

Define

Σ_{lb} is a Horn Lower-bound of Σ

Σ_{ub} is a Horn Upper-bound of Σ

iff Σ_{lb} and Σ_{ub} are sets of Horn clauses and

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

equivalently

$$\Sigma_{lb} \models \Sigma \models \Sigma_{ub}$$

Lower bound = fewer models = logically stronger

Upper bound = more models = logically weaker

Σ_{glb} is a **Greatest Horn lower-bound (GLB)**

Σ_{glb} is a Horn lower-bound, and

No set Σ' of Horn clauses such that

$$\mathcal{M}(\Sigma_{glb}) \subset \mathcal{M}(\Sigma') \subseteq \mathcal{M}(\Sigma).$$

Equivalently: a weakest Horn theory that implies Σ.

Not unique for Σ.

Σ_{lub} is a **Least Horn upper-bound (LUB)**

Σ_{lub} is a Horn upper-bound, and

No set Σ' of Horn clauses such that

$$\mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma') \subset \mathcal{M}(\Sigma_{lub}).$$

Equivalently: strongest Horn theory implied by Σ.

Is **unique** for Σ.

Σ_{glb} and Σ_{lub} are **Horn approximations** of Σ.

Example

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

Horn lower-bound:	$a \wedge b \wedge c$
GLBs:	$a \wedge c$ and $b \wedge c$
Horn upper-bound:	$(\neg a \vee c) \wedge (\neg b \vee c)$
LUB:	c

$$a \wedge b \wedge c \models \underbrace{a \wedge c}_{\text{GLB}} \models \Sigma \models \underbrace{c}_{\text{LUB}} \models (\neg a \vee c) \wedge (\neg b \vee c)$$

Using Approximations for Query Answering

$$\Sigma \models \alpha \ ?$$

- If $\Sigma_{\mathsf{lub}} \models \alpha$ then $\Sigma \models \alpha$.
  (Linear time.)
- If $\Sigma_{\mathsf{glb}} \not\models \alpha$ then $\Sigma \not\models \alpha$.
  (Linear time.)
- Otherwise, use Σ directly.
  (Or return "don't know.")

Queries answered in linear time lead to improvement in **overall** response time to a **series** of queries.

Query Languages

Query language can be more expressive than target language.

Horn Approximations:

Query can be **arbitrary** CNF formula.

Answer in linear time.

Properties of Horn Approximations

LUB can be viewed as an **abstraction**.

Consider background knowledge:

$$\texttt{doctor(X)} \supset \texttt{professional(X)}$$

$$\texttt{lawyer(X)} \supset \texttt{professional(X)}$$

Fact:

$$\texttt{doctor(Jill)} \vee \texttt{laywer(Jill)}$$

LUB is

$$\texttt{professional(Jill)}$$

$$\texttt{doctor(X)} \supset \texttt{professional(X)}$$

$$\texttt{lawyer(X)} \supset \texttt{professional(X)}$$

= abstraction of facts + original background knowledge.

Generalizes (Borgida and Etherington 1989)

GLB can be viewed as a **specialization**.

$\texttt{doctor(Jill)} \vee \texttt{laywer(Jill)}$ becomes $\texttt{doctor(Jill)}$

(in one of the GLBs).

As a **counterexample**.

$\Sigma_{\text{glb}} \not\models \texttt{lawyer(Jill)}$ implies

$\Sigma \not\models \texttt{lawyer(Jill)}$

$\Sigma_{\text{glb}} \models \texttt{doctor(Jill)}$ provides no information.

Compare geometry theorem proving (Gelernter 1969)

As a **positive example**.

Jump to conclusion that $\texttt{Doctor(Jill)}$.

Not sound when used in this way.

GLB is a **set** of models, so generalizes vivid reasoning (Levesque 1986), mental models (Johnson-Laird 1980).

Computing Horn Approximations

Theorem: Let Σ be a set of clauses. The GLB of Σ is consistent iff Σ is consistent. (Similarly for LUB.)

Computing GLB or LUB is **intractable**.

(Provided $P \neq$ NP)

Approximations can be used to check consistency.

View as **compilation process:**

Cost **amortized** over total set of queries.

Computing the GLB

Horn-strengthening:

$p \vee q \vee \neg r$ has Horn-strengthenings

$p \vee \neg r$ and

$q \vee \neg r$

Horn-strengthening of a theory:

$(p \vee q \vee \neg r) \wedge (s \vee t)$ has strengthening

$(p \vee \neg r) \wedge s$

(among others).

Theorem: Each GLB of Σ is equivalent to some Horn-strengthening of Σ.

Algorithm: search space of Horn-strengthenings.

Lemma

Where Σ_H is a Horn theory, and C is any clause:

If Σ_H entails C,

then Σ_H entails some Horn-strengthening of C.

Proof

By completeness of resolution, there is some clause C' that follows from Σ_H by resolution, such that $C' \subseteq C$.

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

Therefore there is some C_H that is a Horn-strengthening of C such that $C' \subseteq C_H \subseteq C$, and so $\Sigma_H \models C_H$.

GLB = Weakest Horn-strengthening

Σ_{glb} is Horn, so by lemma
Σ_{glb} entails **some** Horn-strengthening Σ' of Σ.

$$\Sigma_{\text{glb}} \models \Sigma' \models \Sigma$$

But Σ_{glb} is a **greatest** (weakest) Horn lower-bound.

Therefore, $\Sigma_{\text{glb}} \equiv \Sigma'$.

procedure GLB(Σ)

/* Computes some Horn greatest lower-bound of Σ */

begin

$L :=$ the lexicographically first
Horn-strengthening of Σ

begin 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

Example of GLB Algorithm

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

Horn-strengthenings:

$$L_1 = (\neg a \vee c) \wedge (\neg a \vee b)$$
$$L_2 = (\neg a \vee c) \wedge (\neg a \vee c)$$
$$L_3 = (\neg a \vee c) \wedge (\neg a \vee d)$$

Because

$$L_1 \models L_2$$
$$L_2 \not\models L_3$$

algorithm returns L_2

Properties of the GLB algorithm

- Anytime.

  Algorithm may be stopped at any time to find a lower-bound (not necessary a GLB). Lower-bound improves over time.

- Length of GLB $\leq$ length of original theory.

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)

Example of Computing LUB

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

Resolvents:

$$(1) + (3) = \quad b$$
$$(2) + (3) = \quad a \vee c$$

Answer is

$$(\neg a \vee b) \wedge (\neg b \vee c) \wedge b$$
$$\equiv b \wedge c$$

Algorithm does not resolve $(1) + (2)$

Properties of the LUB algorithm

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

Explosion: an example

Σ is

$$(\texttt{CompSci} \wedge \texttt{Phil} \wedge \texttt{Psych}) \supset \texttt{CogSci}$$

$$\texttt{ReadsMcCarthy} \supset (\texttt{CompSci} \vee \texttt{CogSci})$$

$$\texttt{ReadsDennett} \supset (\texttt{Phil} \vee \texttt{CogSci})$$

$$\texttt{ReadsKosslyn} \supset (\texttt{Psych} \vee \texttt{CogSci})$$

LUB is

$$(\texttt{CompSci} \wedge \texttt{Phil} \wedge \texttt{Psych}) \supset \texttt{CogSci}$$

$$(\texttt{CompSci} \wedge \texttt{Phil} \wedge \texttt{ReadsKosslyn}) \supset \texttt{CogSci}$$

$$(\texttt{CompSci} \wedge \texttt{ReadsDennett} \wedge \texttt{Psych}) \supset \texttt{CogSci}$$

$$\vdots$$

$$(\texttt{ReadsMcCarthy} \wedge \texttt{ReadsDennett} \wedge \texttt{ReadsKosslyn}) \supset \texttt{CogSci}$$

Size LUB $O(2^n)$

No smaller equivalent set of clauses exists!

Shrinking LUB: introduce new concepts

$\Sigma +$ $\texttt{CompSciBuff} \equiv (\texttt{CompSci} \vee \texttt{ReadsMcCarthy})$

$\texttt{PhilBuff} \equiv (\texttt{Phil} \vee \texttt{ReadsDennett})$

$\texttt{PsychBuff} \equiv (\texttt{Psych} \vee \texttt{ReadsKosslyn})$

LUB becomes

$(\texttt{CompSciBuff} \wedge \texttt{PhilBuff} \wedge \texttt{PsychBuff}) \supset \texttt{CogSci}$

$\texttt{CompSci} \supset \texttt{CompSciBuff}$

$\texttt{ReadsMcCarthy} \supset \texttt{CompSciBuff}$

$\texttt{Phil} \supset \texttt{PhilBuff}$

$\texttt{ReadsDennett} \supset \texttt{PhilBuff}$

$\texttt{Psych} \supset \texttt{PsychBuff}$

$\texttt{ReadsKosslyn} \supset \texttt{PsychBuff}$

Size LUB $O(n)$.

- New LUB and original LUB are **equivalent** on queries
  in Lang(Σ).
- New concepts capture what certain pairs of propositions
  have in common.
  E.g., $\texttt{CompSciBuff} \equiv (\texttt{CompSci} \vee \texttt{ReadsMcCarthy})$
- So, new concepts are useful **generalizations** for obtaining
  a tractable approximation of the original theory:
  Forming concepts for fast inference.

QUESTION: does a small (perhaps non-obvious!) representation
of the LUB *always* exist?

What Do We Mean by "Size"?

There are many equivalent clausal theories

$\Sigma \equiv \Sigma'$, yet Σ' exponentially larger

Perhaps: "size" will mean size of

smallest equivalent set of clauses

Not sufficient for proving that something

is **inherently** large: there may be

clever ways to encode a large set of clauses

- Structure sharing,
- Schemas, etc.

Consider then **any** representation of the Horn LUB

that enables polytime inference

The Answer:

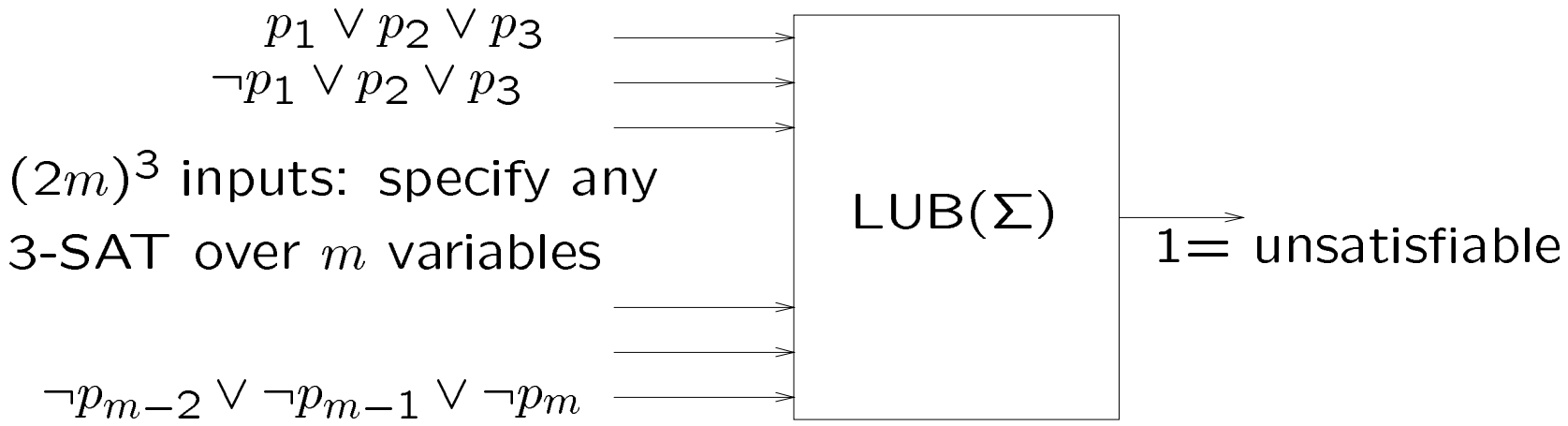
There **do** exist theories whose Horn LUB
is inherently large:

Any representation of the LUB that enables
polynomial time inference is
exponentially larger than the theory!

Proof: Circuit Complexity

Non-uniform P: for every n, there is **some** polysize
circuit for inputs of length n
(equiv: some polytime algorithm)
— same polynomial for all n.

Proof of inherently intractable LUB's:
Can construct a single particular theory
whose LUB can be used to solve 3-SAT
for **any** formula over m variables
(for each m).



Circuit computes "1" iff LUB(Σ) entails that not all the inputs set to "1" can hold simultaneously.

Universal 3-SAT Theory

For 3-SAT formulas over a given set of m variables, define:

$$\Sigma = \bigwedge \{ x \vee y \vee z \vee \neg i_{xyz} \quad | \quad x, y, x \text{ are literals } \}$$

where the "i" variables are new

Any 3-CNF formula over m variables can be specified as a **single** clause made up of the i (input) variables

Example

$(p_1 \vee \neg p_3 \vee p_5) \wedge (\neg p_2 \vee p_3 \vee \neg p_4)$ is unsatisfiable

iff

$(p_1 \vee \neg p_3 \vee p_5 \vee \neg i_{p_1\overline{p_3}p_5}) \wedge (\neg p_2 \vee p_3 \vee \neg p_4 \vee \neg i_{\overline{p_2}p_3\overline{p_4}}) \models$
$\neg i_{p_1\overline{p_3}p_5} \vee \neg i_{\overline{p_2}p_3\overline{p_4}}$

iff

$\mathsf{LUB}(\Sigma) \models \neg i_{p_1\overline{p_3}p_5} \vee \neg i_{\overline{p_2}p_3\overline{p_4}}$

Note: query is Horn, so LUB is complete!

Relation to the Polynomial Hierarchy

Existence of small LUB's for these universal theories would imply NP$\subseteq$non-uniform P

Weaker condition than P=NP:
Polynomial hierarchy collapses to Σ_2

Still considered to be pretty unlikely

Implications

May choose LUB from a **different** tractable class, that **does** guarantee it is small

- k-Horn – bound length of Horn clauses
  O(n^k) max size
- 2-SAT – conjunction 2 literal clauses
  Not Horn – $(p \vee q)$ okay
  O(n^2) max size

Could try to compile to several different tractable classes, pick most powerful class that is small for the particular input theory

GLB and LUB may be different kinds of theories

Extensions

Other target languages:

- 2-SAT (O(n^2) max size).
- k-Horn (O(n^k) max size).
- Clauses **not** containing given set of
  "irrelevant" propositions.
  "Compiling away" parts of original theory.
  Similar to (Subramanian and Genesereth 1987).

First-order source and target languages:

Algorithms may not terminate.

GLB algorithm: interleave comparison of Horn-strengthenings and search.

Other Formalisms: Terminological Logics

Terminological logics (Classic, Brachman et al):

$\mathcal{FL}$: intractable.

$\mathcal{FL}^-$: tractable (no role restrictions).

Compile $\mathcal{FL}$ concepts to $\mathcal{FL}^-$ concepts.

`person`

$\Uparrow$

`(AND person (ALL (RESTR friend male)`

`(AND doctor (SOME specialty)))`

"person whose every male friend is a doctor with a specialty"

$\Uparrow$

`(AND person (ALL friend (AND doctor (SOME specialty))))`

Query Language: Terminological Logics

Recent work by Lenzerini, *et al.* (1991) shows that

Can determine subsumption between

$\mathcal{FL}$ concept and

$\mathcal{FL}^-$ concept

in polynomial time.

Again, query language can be more expressive

than target language.

Does Knowledge Compilation Really Work?

- Can the bounds answer "hard" queries –
  or are such queries easy for original theory?

- Is there **empirical evidence** of savings?
  Classes considered:
  Hard, randomly generated theories
  - relatively unstructured data.
  Planning problems
  - highly structured data.

- Are costs always shifted to compilation time –
  or can the compilation process itself be "paid off"?

Formal Argument for Savings

Trivial case: if KB is inconsistent, compilation
detects this – all queries then are "free".

A more interesting case: Suppose KB is consistent,
logically equivalent to a Horn theory, but
is **not** in Horn form.

- There cannot exist a theorem prover that
  efficiently handles this special case
  (Valiant and Vazirani 1986).
- However, after compilation **all** queries
  can be answered in linear time.

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 computationally hard formulas.

(Mitchell, Selman, and Levesque 1992)

We computed the *unit clause* LUB and GLB

Weaker than the Horn LUB and GLB, but easier to compute and analyze.

Note: unit clause bounds **are** also Horn bounds, but not the best Horn bounds.

Percent of Queries Answered

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.

vars	clauses	size unit LUB	size unit GLB	percent queries answered unit	binary	ternary
75	322	53	71	100%	85%	88%
100	430	57	93	100%	76%	79%
150	645	62	139	100%	66%	66%
200	860	132	188	100%	83%	85%

Execution Time

Implemented query algorithm, using the program
Tableau (Crawford and Auton 93),
a version of the Davis-Putnam procedure,
to handle queries on which bounds fail.

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

vars	clauses	bounds and tableau		tableau only	
		binary	ternary	binary	ternary
75	322	51	48	258	248
100	430	54	45	368	341
150	645	61	59	1286	1084
200	860	55	51	12962	8632

Random Formulas: Summary

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 (approx. 1 hour) completely paid for after 420 3-literal queries – outperformed original goal of simply shifting execution time off-line.

Planning Problems

Domain: Robot moving in a graph-like environment.

(Pollack 1989; Hendler 1990)

Moving to certain nodes consumes resources,
making other nodes inaccessible ("forbidden pairs").

Encoded in the planning as satisfiability framework

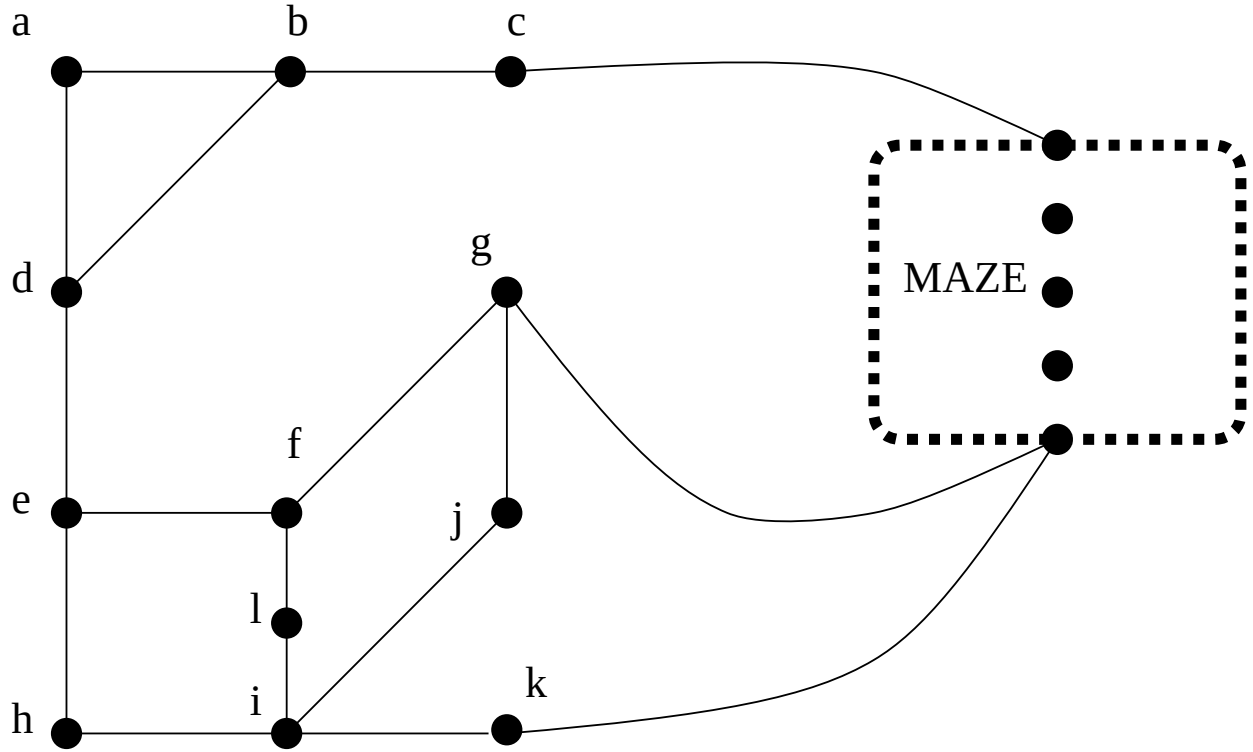
(Kautz and Selman, 1992).

Plans correspond to *models* of the theory.

Planning is NP-complete (*not* just shortest-path).

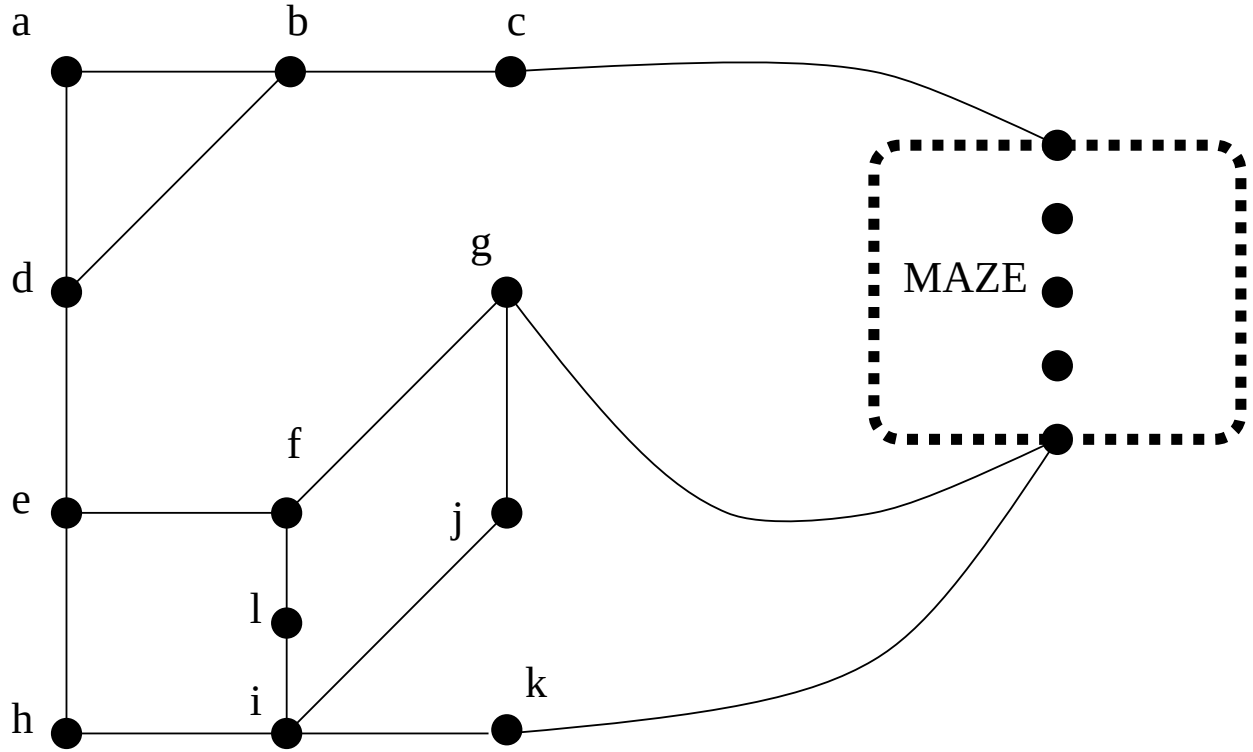
Example: In going from `a` to `g` in at most 5 steps,
can the robot visit `j`?

Answer: No (by length of shortest path).



Example: In going from `a` to `g` in at most 5 steps,
can the robot visit `b`?

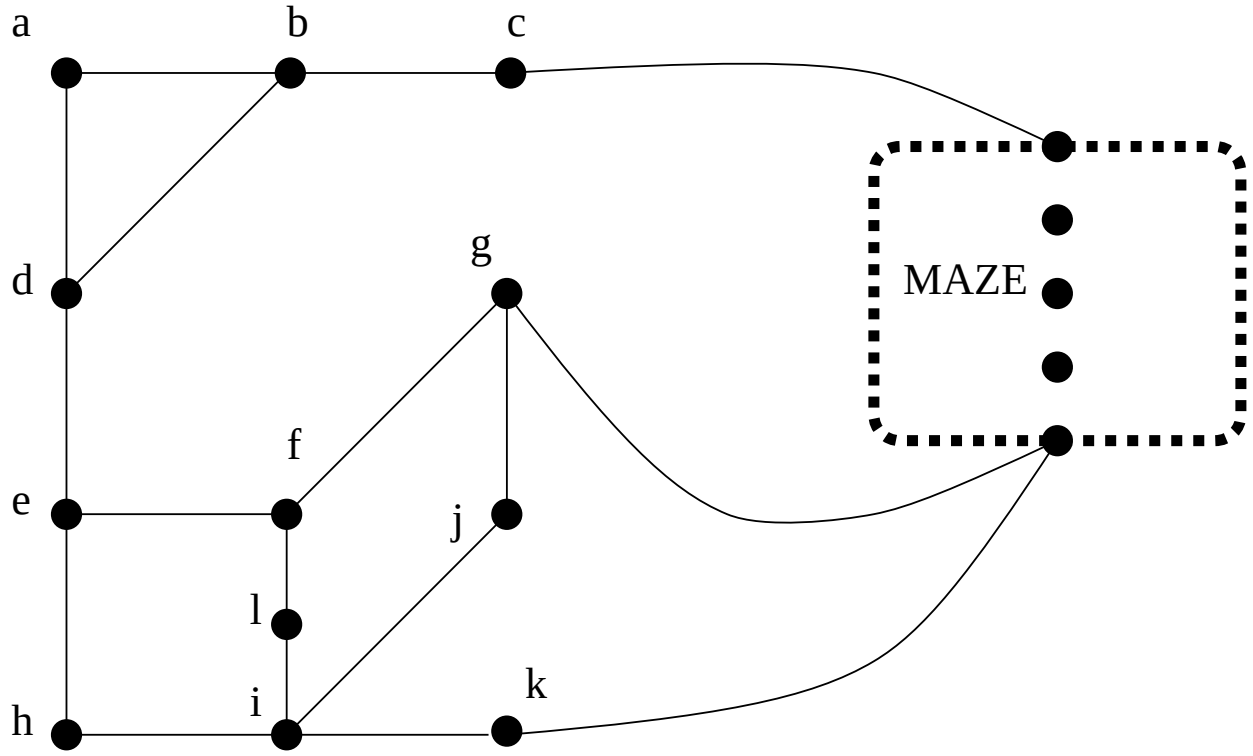
Answer: Yes (a model contains path a-b-d-e-f-g).



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").



Compiling SAT Encoding of Planning Domain

Axioms for mapworld shown in previous figures use

576 variables,

29,576 clauses.

Compiling unit bounds takes 1.2 hours.

Query test sets:

RandBin – 500 random binary queries.

RandEver – 400 random binary queries, restricted to predicates of the form "is the robot *ever* at (a specified) point?"

Hand – 5 hand-constructed "non-obvious" queries.

Planning Results

	RandBin	RandEver	Hand
number of queries	500	400	5
theory Σ only time	**2013**	**8953**	**1071**
KC_Query time	**464**	**3748**	**439**
Σ+LUB time	580	840	6.9
KC using Σ+LUB time	283	617	6.8
bounds only time	**5**	**6**	**1**
number answered by bounds	376	144	2

Times in seconds, on an SGI Challenge.
Theory Σ has 576 variables, 29,576 clauses.
"Σ+LUB time" – using Tableau on conjunction of original theory and its LUB.

Observations

- Basic KC_Query algorithm increased speed by 2X to 4X.
- Similar benefit gained by simply *conjoining* theory and its LUB, and using a complete theorem prover.
- **Best** performance: first test against bounds; if bounds fail, test against theory+LUB — 10X speedup.
- If willing to ignore queries not answered by the bounds – 1000X speedup, handles 36% – 75% of the queries.

  Substitute sensing for theorem proving?

Conclusions

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

- Bounds answer many queries that would be hard to answer with *any* complete theorem prover: success in shifting computational cost off-line.
- Significant speed-up occurs on both unstructured (random) and structured (planning) problems.
- Good performance obtained with unit clause approximations.
- Open question: is additional cost of computing stronger Horn bounds worthwhile?

Summary and Conclusions

Introduced **knowledge compilation:**

A proposal towards obtaining efficient knowledge representation systems.

Features:

- No restrictions on expressiveness of source language.
- Approximations based on two delimiting bounds.

  Generalizes other work on abstraction and "model-based" reasoning.
- Sound and complete inference.
- Efficiency improves over time.
- Generality (any extensional semantics).