Comparative Succinctness of KR Formalisms

Paolo Liberatore

Stupid Example

- x_1=Italian x_2=French, x_3=German
- Ciccio is either Italian or French or German:
  - Explicit: $x_1x_2x_3$, x_1-x_2x_3, x_1x_2-x_3, …
  - Succinct: $x_1 \vee x_2 \vee x_3$
- Explicit: all possible cases;
- Succinct: can be even more intuitive.

Outline

- The problem;
- Direct proofs;
- Compilability proofs;
- Applications of succinctness.

Running example

- Knowledge: a set of modes;
- KB: something representing a set of models
- Language: method for associating a KB to a set of models (and vice versa).
- Example of languages: set of models, 3CNFs, set of terms, formulae, 3CNFs+new variables, default logic, etc.

Representation: Explicit/Succinct

- Explicit: a set of propositional models (tuples of binary values);
- Implicit: a propositional formula.
- Explicit: an ordering of models;
- Implicit: a formula in a language for preference representation.

Expressivity

- Given: two languages LA and LB;
- Question: does every set of models that can be expressed in LA be expressed in LB?
- Not in this talk!

Succinctness

- Given: two languages LA and LB;
- Question: do every set of models that can be expressed in LA be expressed in LB in **polynomial space**?

- This talk is about this.

Results on Succinctness: 2 Kinds

1. Possibilty of polysize translations: ad-hoc proofs (not in this talk);
2. Impossibility:
   - Direct proofs;
   - Proofs based on complexity classes.

Reformulation

The question is the same as:

Can every knowledge base K_1 in LA be translated into a K_2 in LB such that:

- K_1 and K_2 express the same set of modes;
- K_2 is at most polynomially larger than K_1

Direct Proofs: 2 (Sub-)kinds

1. Based only on combinatorial arguments;
2. Based on circuit complexity theory.

Not a theoretical difference.

Notation

- Model: I_1, I_2, I_3,…
- Set of models: S
- Knowledge base: K_1, K_2, …
- Languages: LA, LB

A Trivial Direct Proof

Two languages:

- LA: a KB is a set of complete terms
- LB: a KB is a 3CNF

- Terms: $\{x_1x_2x_3, -x_1x_2x_3, x_1-x_2x_3, \ldots\}$
- 3CNF: $x_1 \vee x_2 \vee x_3$

LB (3CNFs) is "obviously" more succinct.

Considerations

- Most of the languages allow more than one KB to represent the same set of models;
- A language can be short in representing one set of models but longer on another one;
- Size is relevant only for large KB's.

Asymptotic Behavior

- Reduction from LA to LB is possible if:
  - For every S
    - That can be represented in LA in size n
  - It can also be represented in LB in *p(n)*
- Impossibility:
  - Exists $S_1, S_2, \ldots, S_n, \ldots$ such that:
  - S_i can be represented in LA in size n
  - S_i cannot be represented in LB in *p(n)*

Equivalent KB's

- Term: $x_1 \wedge x_2 \wedge x_3$
- 3CNF: $\{x_1 \vee x_2 \vee x_3, \neg x_1 \vee x_2 \vee x_3, x_1 \vee \neg x_2 \vee x_3, \ldots\}$

Sets of terms are more succinct than 3CNFs?

- Equivalent 3CNF: $\{x_1, x_2, x_3\}$;
- Always consider the most succinct KB's!

Example

The proof for terms vs. 3CNFs:

- $\{\{x_1,x_2,x_3\}\}$ is a specific set of models
- $\{\{\{x_1 \wedge x_2 \wedge \ldots \wedge x_n\}\} \mid n>0\}$ is a set of sets

3CNFs can be more succinct than sets of terms: proved by the second, not the first.

Specific Sets

- Incomparable languages:
  - LA: S is short but R is large;
  - LB: S is large and R is short.
- Comparable: every S that is short in LA is short in LB as well.

Circuit Complexity

- Classes within P;
- Non-conditioned results.

A useful result:

- PARITY is not in AC^0
- Meaning: no polynomial-size CNF formula represents the set of all models with an even number of 1's.

A Language

- Language of 3CNFs with new variables
- KB=(F,X,Y) where:
  - F: a 3CNF formula on variables X∪Y (disjoint)
- Represents sets of models on variables X
- I (a model on variables X) is in the set represented by KB=(F,X,Y) if there exists a model J on variables Y such that I∪J is a model of F

Proofs Using Complexity Classes

- Largest part of the talk
- Idea: given a problem on S that
  - is hard if S is expressed in LA
  - is simple if S is expressed in LB

translating from LA to LB must be difficult!

(otherwise, solve by first translating!)

Application of PARITY

- LA=language of 3CNFs;
- LB=language of 3CNFs with new variables.

We can use PARITY to prove that LB is more succinct than LA

More Notations…

- I∈S means that I is a model of S
- I∈KB, where KB is a knowledge base, means that KB represents a set of models that contains S

Checking I∈KB is a decision problem.
Can be represented by a set: A={(K,I)|I∈K}

PARITY in Action

S_n=all models of n variables with an even number of 1's

- In LA: not in polynomial space;
- In LB: since parity can be checked in polynomial time, there exists a circuit (a specific kind of formulae with new variables) that represents S_n in polynomial space.

Easy Result

- I∈S is a polynomial-time problem;
- I∈KB can NP-hard:
  - It is if KB is in the language of 3CNFs+new variables.

Have we proved that the language of 3CNFs+new variables is more succinct than the explicit representation?

NO!

Hardness and Size I

- Hardness: how long does it take;
- Succinctness: how much space is needed.

Referred to a language:

- Hard: takes a long time to translate;
- Succinct: translating produce large result.

Compilability

Digression (>10 slides!);

- How hard is a problem if part of its data can be preprocessed?
- Example: in diagnosis, we have:
  - the description of the system to diagnose;
  - the specific faults.
- They do not have the same status.

Hardness and Size II

- Languages for representing a single bit:
  - LA: explicit representation (0 or 1);
  - LB: a bit is represented by a Turing machine:
    - the machines that always terminate represent 1;
    - the others represent 0.
- Translating from LB to LA is undecidable.

Is LB more succinct?

Assumptions on Preprocessing

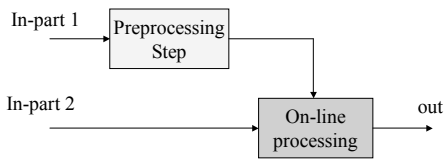
- Solving is done in two steps:
  - First preprocess one part of the input only;
  - Then, solve the problem.
- The first phase (the preprocessing step):
  - Can take arbitrarily long time;
  - Must produce a polynomially-sized result.

Hardness≠Size

- Fact:
  - Translating from LB to LA is hard (undecidable in this case!);
  - Translation result is polynomially-sized.
- Consequence:
  - Hardness cannot be used to compare succinctness.

(btw: both 0 and 1 have short TM representation: LA and LB are succinctly equivalent)

Preprocessing, Pictorially



Classes of Compilability

- Complexity of the on-line part;
- The complexity of the preprocessing step is not counted.

- Complexity: P and NP.
- Compilability: ~>P and ~>NP.

The function *f*

Is the in/out function of the preprocessing step

- Its computation is not bounded on time;
- Its result must be of polynomial size w.r.t. the size of its argument.

Formally: *f* is **polysize** if there exists a polynomial p such that, for every string *x*, it holds $|f(x)|<p(|x|)$

Classes: Formal Definition

- A problem is a set of pairs of strings;
  - E.g, A={(*x*,*y*)}
- Solving=telling whether $(x,y)\in A$ for a given pair of strings (*x*,*y*)

Idea: *x* is the part we can preprocess;
Usual formalization of decision problems.

Must f be computable?

Depending on what we try to prove:

- That a problem is in ~>P; reasonable to assume that *f* is computable;

- That a problem **is not** in ~>P: stronger results if *f* is not bounded.

Formal definition II

- Class ~>P: is a set of problems A={(*x*,*y*)}
- $A\in$ ~>P if there exists:
  - Problem $B\in P$
  - Function *f* from strings to strings (see below!)
- Such that:
  - $(x,y)\in A$ if and only if $(f(x),y)\in B$

Back to Succinctness…

- The question was: given K_1 in LA, is there any equivalent K_2 in LB that is (at most) polynomially larger?

- Equivalence means: $I\in K_1$ iff $I\in K_2$;
- Question, reformulated: solve the problem $I\in K_1$ by preprocessing K_1 into K_2.

Complexity and Compilability

- Problem A is $I \in K_1$;
- Problem B is $I \in K_2$;
- Complexity of B: polynomial;
- If every K_1 in LA has an equivalent K_2 in LB of polynomial size, then:
- $A \in \sim>P$

  (f=the function that gives K_2 given K_1)

How to prove non-membership?

- Membership to ~>P: no general method;
- Non-membership: proofs based on hardness

- Seen: definition of ~>P is based on P;
- Now: definition of ~>NP based on NP;
- Generalization to an arbitrary class of problems C.

Why?

- Facts:
  - $I \in K_1$ is equivalent to $I \in K_2$;
  - K_1 can be translated into K_2 (not in P!)
  - $I \in K_2$ is in P
  - f defined as $f(K_1)=K_2$ is a polysize function
  - $I \in K_1$ iff $I \in K_2$
- Consequence:
  - Solving $I \in K_1$ is in ~>P

Compilability Classes

- Replace P with another class C everywhere:
  - $A \in \sim>C$ if there exists B and f such that:
  - $B \in C$
  - $(x,y) \in A$ iff $(f(x),y) \in B$
- Function f is polysize:
  - Result is at most polynomially larger than argument.

So What?

The other way around:

- Prove that $I \in K_1$ is not in ~>P
- Conclude that K_1 cannot be translated into a polynomially-sized K_2

This is a method for obtaining negative results (impossibility of polysize translations).

Compilability-Hardness

- Based on polynomial reductions;
- Direct definition of hardness not useful;
- Classes ||~>C: the preprocessing step can use the first part of data and the size of the second part;
- The corresponding hardness is useful.

Monotonic Reductions

- Proving ||~> hardness is… hard;
- Sufficient conditions:
  - Monotonic reductions;
  - Representative equivalence.
- Only sufficient;
- Usually work.

Operatively…

- Usually, A is already known NP-hard;
- Polynomial-time reduction from 3sat to A known;
- Often, does not satisfy the condition of representative equivalence.

In such cases: find a new reduction.

Monotonic Reductions: the Base

- Problem A={(x,y)} is NP-hard;
  - Complexity, not compiability;
- Means:
  - there exists two polynomial functions r,h;
  - F is sat iff (r(F)),h(F))∈ A
- How can A be proved ||~>NP-hard?

Reduction: Guideline I

- A is the problem of checking whether a model I satisfied a knowledge base K;
- A={(K,I)|I is a model of K}
- Reduction from 3sat to A:
- F is safisfiable iff I is a model of K
- If K depends only on the number of variables of F the reduction is monotonic.

Monotonic Reductions

r, h: polynomial reduction from 3sat to A

- For every two 3CNF formulae F and G that:
  - Have the same variables;
  - F⊆G (i.e., G has some clauses more than F)

If: (r(F),h(F))∈ A iff (r(G),h(F))∈ A

Then: problem A is ||~>NP-hard.

[there is no typo in this slide]

Reduction: Guideline II

F=variables+structure (clauses)

- Variables of F ➔ K
- Whole formula F ➔ I

How can this be done?

- F is a 3CNF of n variables
- Given n variables, there are only $O(n^3)$ possible clauses of three variables.

Reduction: Guideline III

- F ➔ G=$\{(v_i \rightarrow c_i) | c_i \in C_n\}$
- v_i are new variables
- C_n=set of all 3-clauses on the same variables of F
- F is "almost" equivalent to G$\cup\{v_i | c_i \in F\}$

Reduce:

- G➔K
- $\{v_i | c_i \in F\}$ ➔I

Easier to reduce a set of variables to a model.

Does it always work?

- Sufficient condition;
- G to K and $\{v_i | c_i \in F\}$ to I is hard sometimes.

Intuitive meaning, based on structures.

Reduction: Example

- Language of 3CNF with new variables;
- Is NP-hard; by reduction from 3sat:
  - 3CNF formula F on variables X is sat if and only if the empty model is a model of (F,∅,X)
- This reduction is not monotonic.

Generalization

Often, we have:

- A collection of objects (e.g., propositional variables);
- These objects form structures (e.g., clauses, defaults, etc.)
- K is a collection of these structures.

Idea: use subcase with few possible structures.

A Monotonic Reduction

- F ➔ G=$\{(v_i \rightarrow c_i) | c_i \in C_n\}$ where:
  - C_n=all clauses of three variables over the same variables of F
- F is sat iff G$\cup\{v_i | c_i \in F\}$ is sat;

Consequence:

- F is sat iff $\{v_i | c_i \in F\}$ is a model of G;
- Is a monotonic reduction.

Application I

- Object: nodes;
- Structures: edges;
- Knowledge base: graph.

n nodes: at most n^2 possible edges.

Application 2

- Object: variables;
- Structures: formulae and defaults;
- Knowledge base: default theory.

Limit to the case of defaults containing only a fixed number of variables.

Time/Space Tradeoff

- LA is compilability-hard➔it is succinct
- LB is compilability-simple➔is not succinct

Compilability hardness prove succinctness.

Note: a language that is hard but not compilability hard is both hard and not succinct.

Intuition

- What these reductions prove?
- F contains two pieces of information:
  - The number of variables;
  - The clauses.
- We reduce the clauses to I and the number of variables to K;
- The complexity is in I, not in K;
- Preprocessing K is useless.

Knowledge Bases

- Structures that represent knowledge;
  - So far: knowledge=set of models;
- Could also be:
  - Knowledge=set of propositional formulae;
  - Knowledge=ordering of models;
  - ???

Preprocessing and Succinctness

- A=checking whether a model is a model of a knowledge base in language LA
- B=the same for LB

If A is ||~>NP-hard and B is in ~>P;

There exists knowledge bases in LA that cannot be polynomially expressed in LB.

References

- Cadoli et al. Preprocessing of intractable problems, I&C 176(2), 2002.
- Liberatore, Monotonic reductions, representative equivalence, and compilation of intractable problems, JACM 48(6), 2001.
- Cadoli et al. Space efficiency of propositional knowledge representation formalisms, JAIR 2000.