Efficient Decision Procedures for Heaps Using STRAND

P. Madhusudan and Xiaokang Qiu

University of Illinois at Urbana-Champaign

SAS'11 (Static Analysis Symposium)

Decidable logics on Heaps

Decision procedures for heaps is hard:

- Heaps have unboundedly many nodes
- Logics over heaps must have some form of quantification
- It is even harder to reason with data: standard techniques like Nelson-Oppen fail to combine theories of heap structure and data

Decidable logics on Heaps

Decision procedures

- Only heap structure: PALE [KS93], TASC [HIV10]
- With data: HAVOC [LQ08], CSL [BDES09] Neither can express binary search trees
- STRAND [MPQ-POPL11]
  - Powerful logic combining structure and data
  - Can express properties of trees: BST, etc.
  - Decidable semantic and syntactic fragments

STRAND Logic [MPQ-POPL11]

We concentrate on the syntactic decidable fragment of STRAND and build better decision procedures.

STRAND logic [MPQ11] is parameterized by

- a class of recursively defined data-structures *R*, defined using Monadic Second-Order (MSO) relations
- A data-logic *D*, with a decidable quantifier-free theory
- Results in this paper hold for arbitrary *R* and arbitrary *D*

In this talk, we fix

R: class of binary trees with a unique NIL node

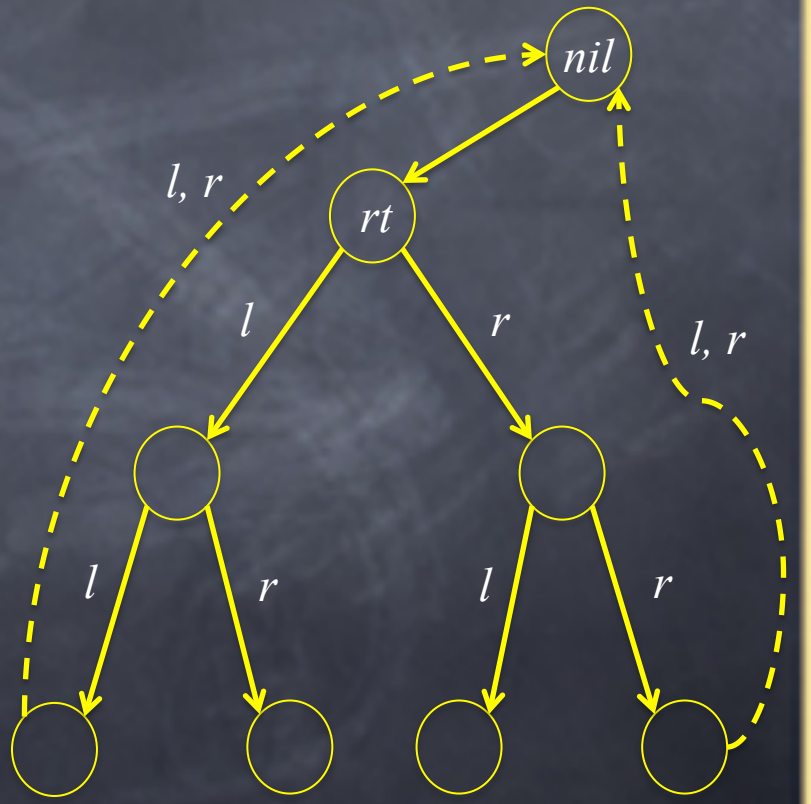
D: linear integer arithmetic.

Binary Trees with a NIL node

Elastic and non-elastic relations

- non-elastic relations: left-child/right-child
- elastic relations: left-desc/right-desc

- For arbitrary recursively defined data-structure, there is a formal definition of elasticity; also elasticity can be decided.



Syntax of STRAND

Key restriction: Non-elastic relations (left/right-child) can only relate existentially quantified variables.

$$
\begin{array}{lll}
\text{Formula} & \psi ::= & \exists \vec{x}\, \forall \vec{y}\, .\, \varphi \\
\text{QFFormula} & \varphi ::= & e_1 < e_2 \mid e_1 = e_2 \mid root(v) \mid nil(v) \mid v = v' \\
 & & \mid left\,desc(v, v') \mid right\,desc(v, v') \\
 & & \mid left\,child(x, x') \mid right\,child(x, x') \\
 & & \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \\
\text{DataExpr} & e ::= & data(v) \mid c \mid e_1 + e_2 \mid e_1 - e_2
\end{array}
$$

$$
\begin{array}{llll}
\exists\text{DVar} & x \in Loc & \text{DVar} & v ::= x \mid y \\
\forall\text{DVar} & y \in Loc & \text{DataConst} & c \in Int
\end{array}
$$

$$Example : \forall y.(\neg nil(y) \rightarrow data(y) = 1)$$

STRAND for Program Verification

- Expressiveness
  - Can express correctness properties of doubly-linked lists, Cyclic lists, Binary search trees…
- Program verification

  Hoare-triple Validity w/ pre-cond and post-cond written in universal/existential fragment of STRAND

  ➔ Verification conditions in STRAND
- Decidability of satisfiability
  - Precise, Terminating, Provides counterexamples

Known Decision Procedure [MPQ11]

$$\psi \;=\; \exists \vec{x}\, \forall \vec{y}\, .\, \varphi(\vec{x}, \vec{y})$$

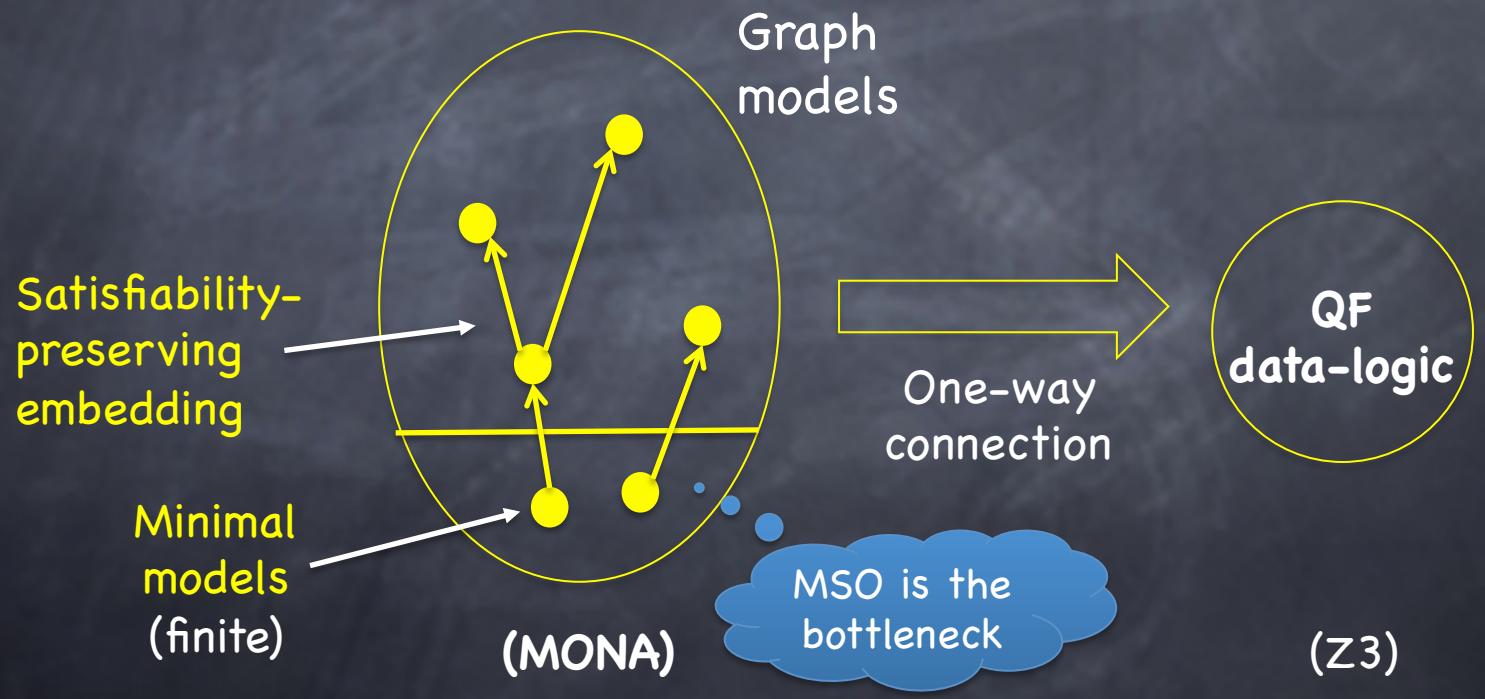
Key ideas

- A STRAND formula is satisfied iff there is a model of size *B*, where *B* can be computed.
- Bound *B* is non-trivial to compute; requires solving MSO formulas on trees
- Data-agnostic satisfiability-preserving embedding "→" :

  { *M* → *M′* } then { *M′* sat Ψ => *M* sat Ψ }
- Use pure MSO over trees capture the set of minimal models wrt embedding and use it to compute *B*

Known Decision Procedure [MPQ11]

Based on the data-agnostic Small Model Property



Where is the bottleneck?

$$\psi \;=\; \exists \vec{x}\; \forall \vec{y}\; .\; \varphi(\vec{x}, \vec{y})$$

T is a Minimal graph model ⇔
There is no sub-model *S* such that
S satisfies Ψ whenever *T* does (data-agnostically!)

There is no sub-model *S*

$$MinModel = \neg \exists X. \Big(\; Submodel(X) \; \wedge$$

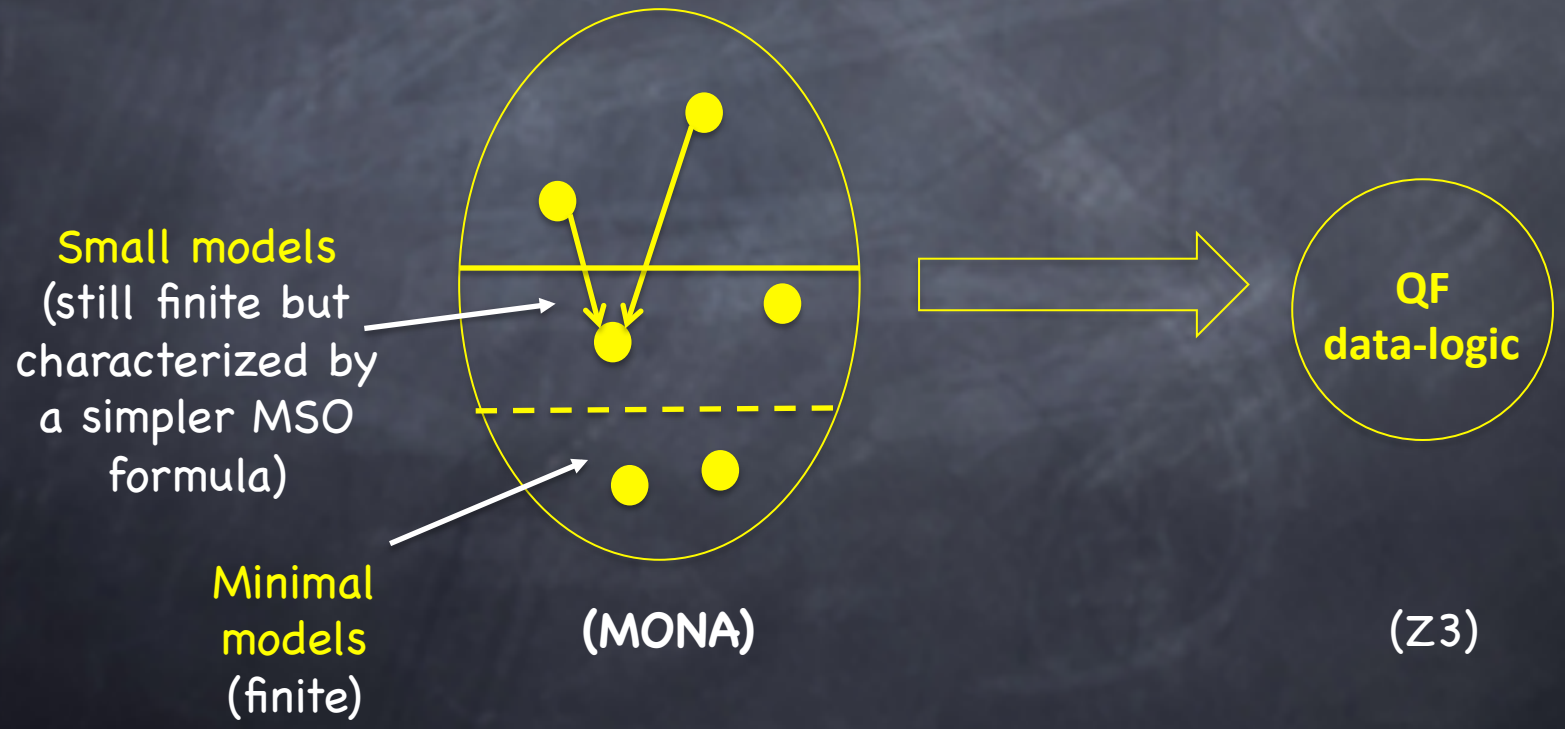
$$\forall \vec{y} \in X \; \forall \vec{p} \; \Big(\widehat{\varphi}(\vec{y}, \vec{p}) \Rightarrow tailor_X\big(\widehat{\varphi}(\vec{y}, \vec{p})\big) \Big) \; \Big)$$

T sat φ

S sat φ

Our Contribution

- Key idea of this paper: A new decision procedure that computes a conservative approximation of minimal models using much simpler formulas (some formulas are solved 1000x faster)



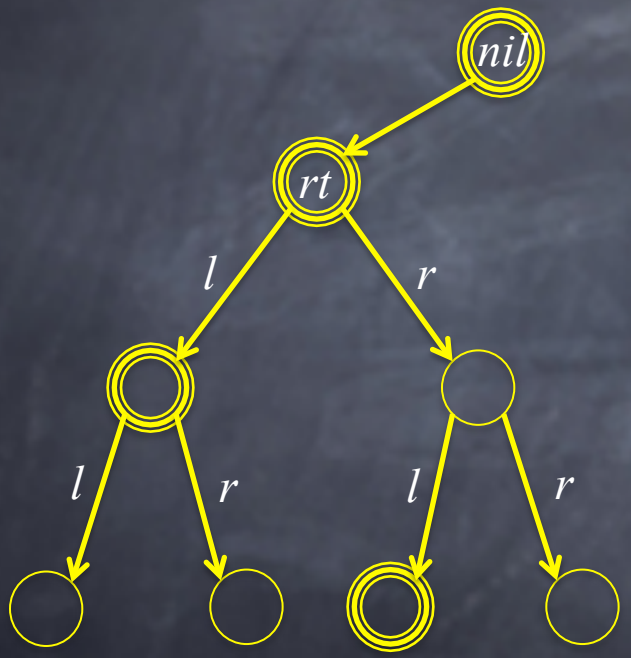
Outline

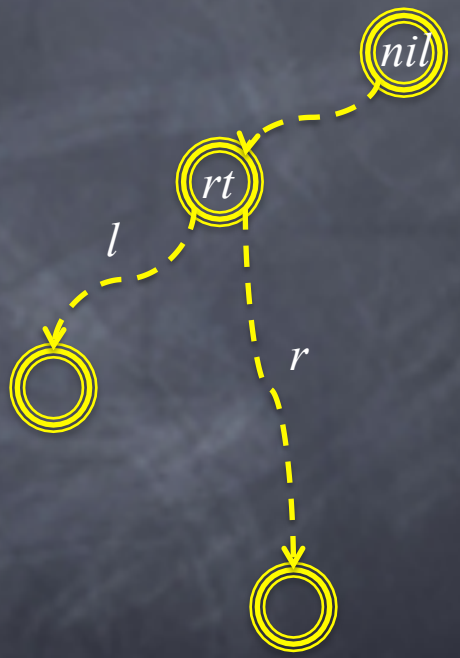
- Sub-models
- Elasticity of relations
- New decision procedure
- Comparison with earlier decision procedure

Sub-models

- The notion of minimal/small models requires the definition of sub-models (sub-trees)
- A sub-tree can be constructed from a valid subset of vertices of a tree
  - Unique closest left/right descendent
  - Vertex labels are preserved

Sub-models: Example

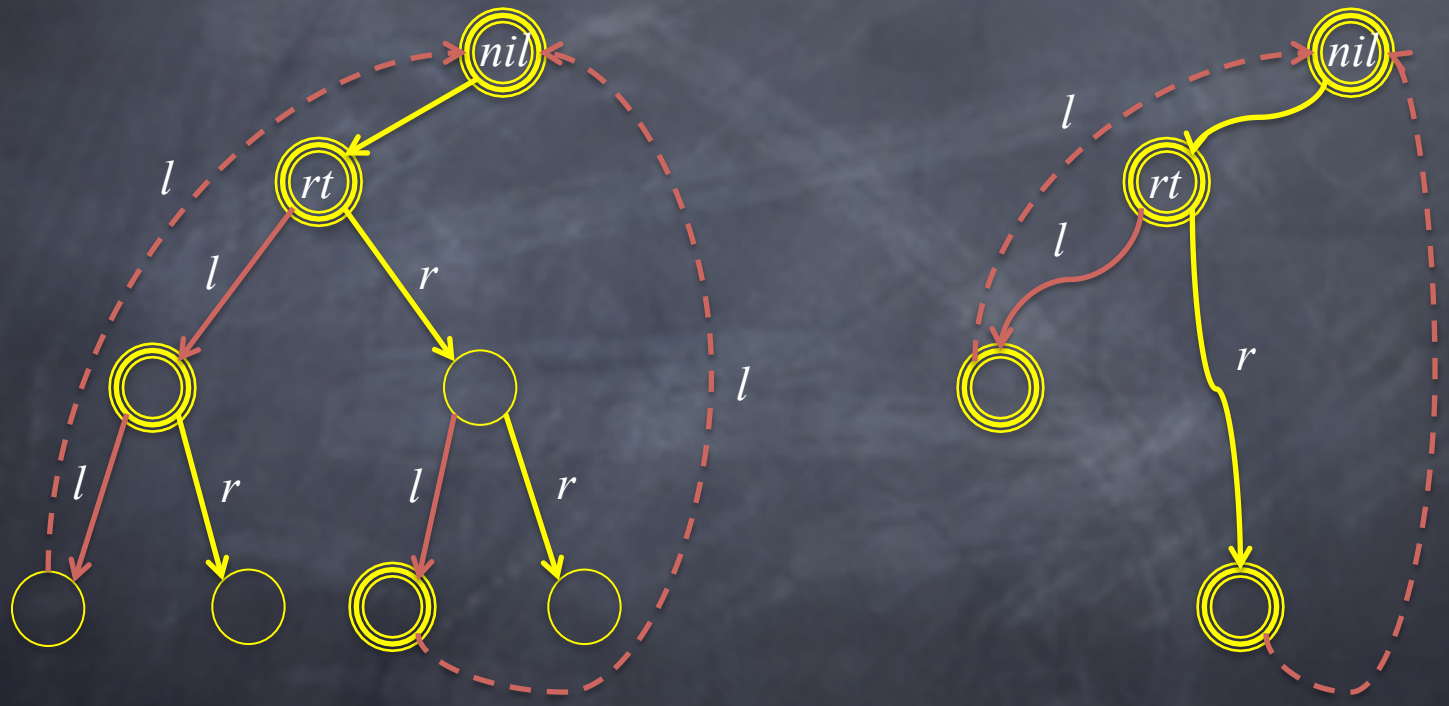




Outline

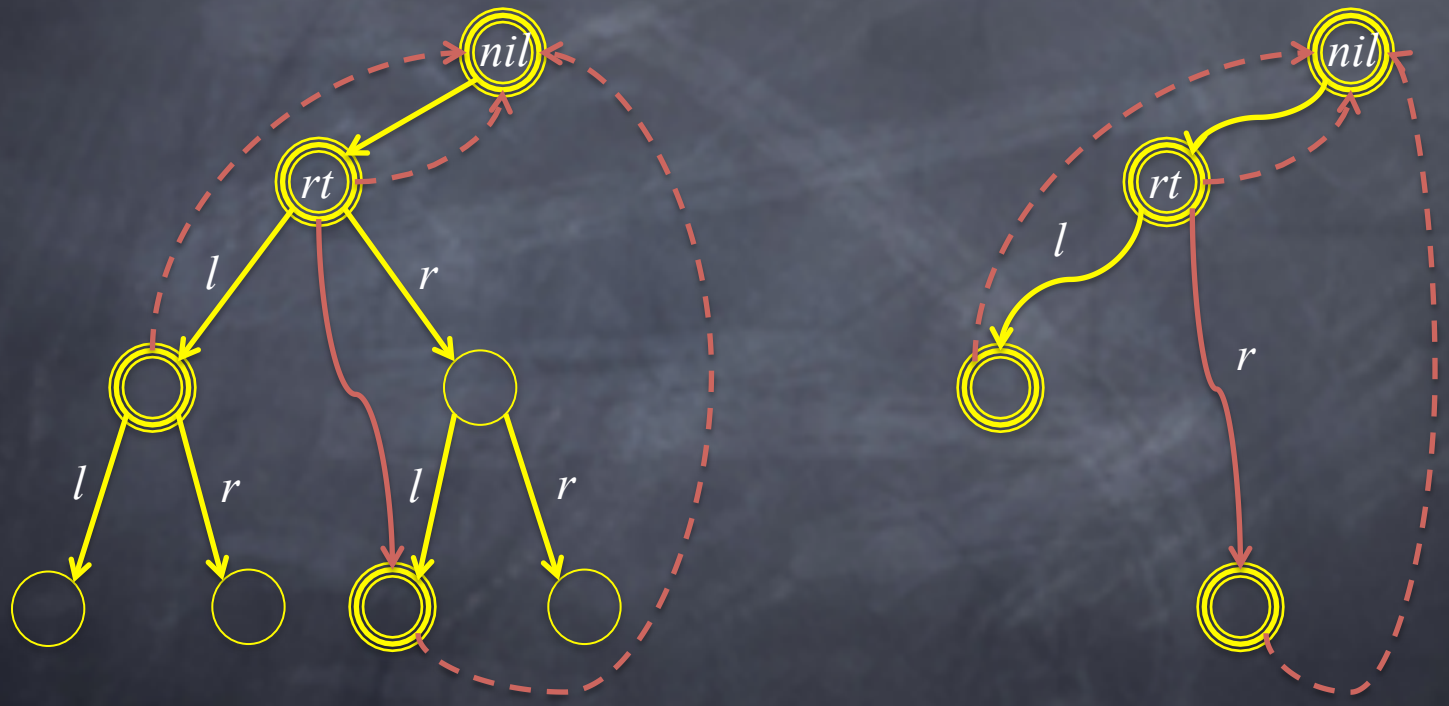
- Sub-models
- Elasticity of relations
- New decision procedure
- Comparison with earlier decision procedure

Non-elastic Relation: *left-child*



Not always preserved in sub-models

Elastic Relation: *right-desc*



Always preserved in sub-models

Outline

- Sub-models
- Elasticity of relations
- New decision procedure
- Comparison with earlier decision procedure

Small Models

$$\psi \;=\; \exists \vec{x}\, \forall \vec{y}\, .\, \varphi(\vec{x}, \vec{y})$$

A small model T does not contain a sub-model evaluating ψ in the same way.

T is a small model $\Leftrightarrow$

There is no sub-model S such that

for every non-elastic relation possibly appearing in φ, it holds in T iff it holds in S.

(Elastic relations are always preserved.)

Represent Small Models in MSO

There is no sub-model *S*

$$SmallModel(\vec{x}) \equiv \neg\exists X.\Big(Submodel(X) \wedge \bigwedge_{x\in\vec{x}}(x \in X) \wedge \bigwedge_{r\in NE, x,x'\in\vec{x}} \Big(r(x,x') \Leftrightarrow tailor_X\big(r(x,x')\big)\Big)\Big)$$

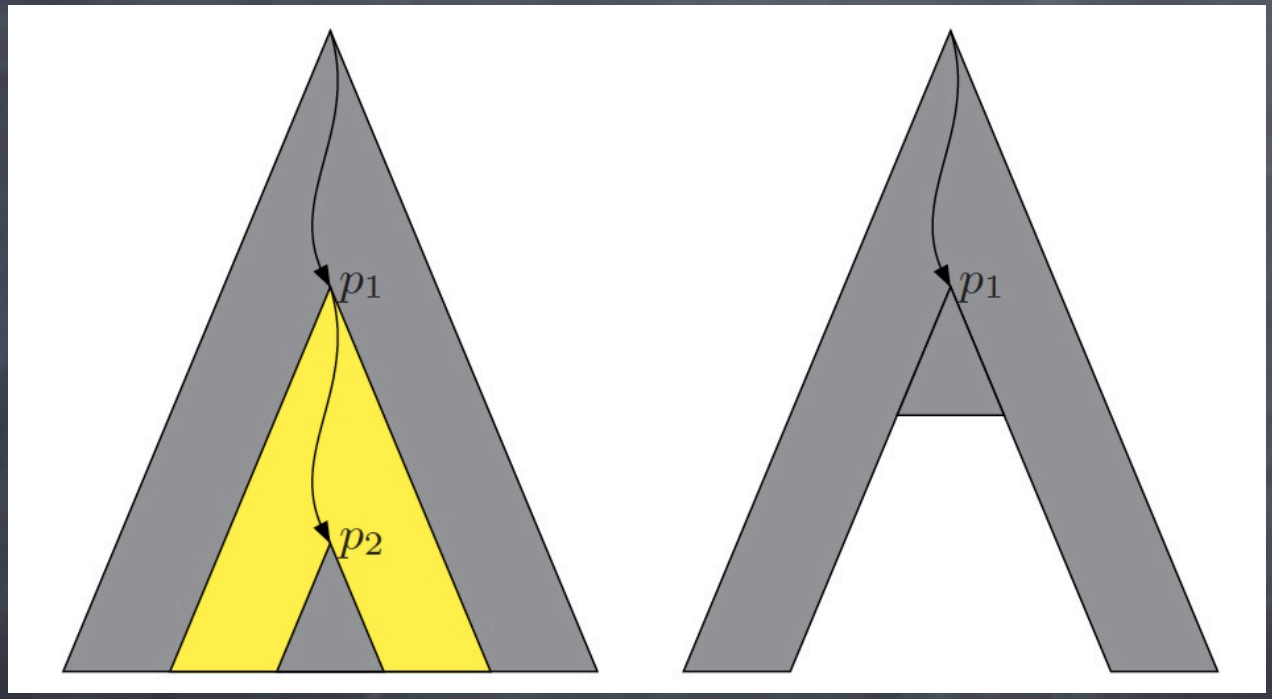
For every non-elastic relation, every possible pairs of variables

T sat *r* iff *S* sat *r*

Two Main Technical Results

- Finiteness
  - For any recursively defined data-structure and any set of existential variables, the number of small models is finite.
- Proof Sketch
  - Let T be a model of ψ
  - Using the classic logic-automata connection, construct a deterministic bottom-up tree automaton for every non-elastic relation r, it exactly accepts/rejects $r(x, x')$ iff T does/doesn't satisfy $r(x, x')$
  - The product of these automata accepts T
  - If T is large enough, we can pump it down

Two Main Technical Results



T with the valid subset *X* (shaded dark)

A smaller model constructed by X

Two Main Technical Results

- Equisatisfiability
  - If a STRAND formula ψ is satisfiable, it is satisfied by a small model.
- Proof Sketch
  - Let T be the model of ψ with least number of nodes, then T is a small model.
  - Otherwise a sub-model of T also satisfies ψ
    (by induction on the structure of ψ).

New Decision Procedure

Is ψ satisfiable?

1. Compute a tree automaton accepting the set of all small models. (MONA)

2. Compute the maximum height h of the small models.

3. Query the data-solver as to whether there is a model of height up to h with data that satisfies ψ. (Z3)

Outline

- Sub-models
- Elasticity of relations
- New decision procedure
- Comparison with earlier decision procedure

Theoretical Comparison

$$MinModel = \neg\exists X.\Big(Submodel(X) \wedge \forall \vec{y} \in X \; \forall \vec{p} \Big(\widehat{\varphi}(\vec{y},\vec{p}) \Rightarrow tailor_X\big(\widehat{\varphi}(\vec{y},\vec{p})\big)\Big)\Big)$$

$$SmallModel(\vec{x}) \equiv \neg\exists X.\Big(Submodel(X) \wedge \bigwedge_{x\in\vec{x}} (x \in X) \wedge \bigwedge_{r\in NE, x, x'\in\vec{x}} \Big(r(x,x') \Leftrightarrow tailor_X\big(r(x,x')\big)\Big)\Big)$$

Program	Verification condition	Minimal Model		Small Model		Data Constraint Solving Old/New Time (s)
		Max. BDD Size	Time (s)	Max. BDD Size	Time (s)	
Sorted-list-search	before-loop	10009	0.34	540	0.01	
	in-loop	17803	0.59	12291	0.14	-
	after-loop	3787	0.18	540	0.01	-
Sorted-list-insert	before-head	59020	1.66	242	0.01	0.02/0.02
	before-loop	15286	0.38	595	0.01	-
	in-loop	135904	4.46	3003	0.03	-
	after-loop	475972	13.93	1250	0.01	0.02/0.03
Sorted-list-insert-error	before-loop	14464	0.34	595	0.01	0.02/0.02
Sorted-list-reverse	before-loop	2717	0.24	1155	0.01	-
	in-loop	89342	2.79	12291	0.14	-
	after-loop	3135	0.35	1155	0.01	-
bubblesort	loop-if-if	179488	7.70	73771	1.31	-
	loop-if-else	155480	6.83	34317	0.48	-
	loop-else	95181	2.73	7017	0.07	0.02/0.04
bst-search	before-loop	9023	5.03	1262	0.31	-
	in-loop	26163	32.80	3594	2.43	0.02/0.11
	after-loop	6066	3.27	1262	0.34	-
bst-insert	before-loop	3485	1.34	1262	0.34	-
	in-loop	17234	8.84	1908	1.38	-
	after-loop	2336	1.76	1807	0.46	-
left-rotate	bst-preserving	1086	1.59	1510	0.48	0.05/0.14
Total			98.15		7.99	0.15/0.36

http://www.cs.uiuc.edu/~qiu2/strand/

Pre-compute the bound

- Given a recursively defined data-structure, the bound for small models is only determined by the number of existentially quantified variables.
- Fix a recursively defined data-structure, a lookup table can be pre-computed for up to 5/10 variables and be reused everywhere!
- We can even establish these bounds analytically, and the structural phase can be completely avoided!
- Example: In the binary tree example, for *n* variables, a small model is of height up to *2n*.

Conclusion

- The earlier decision procedure for syntactic decidable fragment of STRAND computes minimal structural models in a completely data-logic agnostic manner.
- Our new decision procedure gives a way of computing small structural models that is even agnostic to the STRAND formula.
- Much simpler in theory, and much faster in practice.
- Thank you for your attention!