Complexity of Decision Problems in Computational Logic

Vincent Jugé
X 2006

William Marsh Rice University
Computer Science Department

August 28, 2009

Is the equivalence of two algorithms decidable ?

- In the general case, **no** :
  - Undecidability of the Halting Theorem.
- In some particular cases, **yes** :
  - $f : n \to 0$
  - $f : n \to 1$

How can we evaluate the computational complexity of a problem ?

- Look for lower bounds :
  - Evaluate the computational complexity of the problem for particular instances.
- Look for upper bounds :
  - Find an algorithm solving the problem and evaluate its complexity.

What is a Datalog program ?

- A set of Horn rules without function symbols.
- A **goal** predicate.

Example of Datalog program

Program Π :

- $\mathbf{Z}(X) \Rightarrow \mathbf{E}(X)$
- $\mathbf{E}(X) \wedge \mathbf{S}(X, Y) \Rightarrow \mathbf{O}(Y)$
- $\mathbf{O}(X) \wedge \mathbf{S}(X, Y) \Rightarrow \mathbf{E}(Y)$
- $\mathbf{O}(X) \Rightarrow \overline{\mathbf{O}}(X)$
- Goal predicate : $\overline{\mathbf{O}}$.

Datalog programs work on deductive databases

- $D = (\{0,1,2,3\}, \{\mathbf{Z}(0), \mathbf{S}(0,1), \mathbf{S}(1,2), \mathbf{S}(2,3)\})$
- $\Pi(D) = (\{0,1,2,3\}, \{\mathbf{Z}(0), \mathbf{S}(0,1), \mathbf{S}(1,2), \mathbf{S}(2,3)\})$

Datalog programs work on deductive databases

- $D = (\{0,1,2,3\}, \{\mathbf{Z}(0), \mathbf{S}(0,1), \mathbf{S}(1,2), \mathbf{S}(2,3)\})$
- $\Pi(D) = (\{0,1,2,3\}, \{\mathbf{Z}(0), \mathbf{S}(0,1), \mathbf{S}(1,2), \mathbf{S}(2,3), \mathbf{E}(0)\})$

Datalog programs work on deductive databases

- $D = (\{0,1,2,3\}, \{\mathbf{Z}(0), \mathbf{S}(0,1), \mathbf{S}(1,2), \mathbf{S}(2,3)\})$
- $\Pi(D) = (\{0,1,2,3\}, \{\mathbf{Z}(0), \mathbf{S}(0,1), \mathbf{S}(1,2), \mathbf{S}(2,3), \mathbf{E}(0), \mathbf{O}(1)\})$

Datalog programs work on deductive databases

- $D = (\{0,1,2,3\}, \{\mathbf{Z}(0), \mathbf{S}(0,1), \mathbf{S}(1,2), \mathbf{S}(2,3)\})$
- $\Pi(D) = (\{0,1,2,3\}, \{\mathbf{Z}(0), \mathbf{S}(0,1), \mathbf{S}(1,2), \mathbf{S}(2,3),$
  $\mathbf{E}(0), \mathbf{O}(1), \overline{\mathbf{O}}(1), \mathbf{E}(2)\})$

Datalog programs work on deductive databases

- $D = (\{0,1,2,3\}, \{\mathbf{Z}(0), \mathbf{S}(0,1), \mathbf{S}(1,2), \mathbf{S}(2,3)\})$
- $\Pi(D) = (\{0,1,2,3\}, \{\mathbf{Z}(0), \mathbf{S}(0,1), \mathbf{S}(1,2), \mathbf{S}(2,3),$
  $\mathbf{E}(0), \mathbf{O}(1), \overline{\mathbf{O}}(1), \mathbf{E}(2), \mathbf{O}(3)\})$

Datalog programs work on deductive databases

- $D = (\{0,1,2,3\}, \{\mathbf{Z}(0), \mathbf{S}(0,1), \mathbf{S}(1,2), \mathbf{S}(2,3)\})$
- $\Pi(D) = (\{0,1,2,3\}, \{\mathbf{Z}(0), \mathbf{S}(0,1), \mathbf{S}(1,2), \mathbf{S}(2,3),$
  $\mathbf{E}(0), \mathbf{O}(1), \overline{\mathbf{O}}(1), \mathbf{E}(2), \mathbf{O}(3), \overline{\mathbf{O}}(3)\})$

Datalog programs work on deductive databases

- $D = (\{0,1,2,3\}, \{\mathbf{Z}(0), \mathbf{S}(0,1), \mathbf{S}(1,2), \mathbf{S}(2,3)\})$
- $\Pi(D) = (\{0,1,2,3\}, \{\mathbf{Z}(0), \mathbf{S}(0,1), \mathbf{S}(1,2), \mathbf{S}(2,3),$
  $\mathbf{E}(0), \mathbf{O}(1), \overline{\mathbf{O}}(1), \mathbf{E}(2), \mathbf{O}(3), \overline{\mathbf{O}}(3)\})$
- Solutions of $\overline{\mathbf{O}}(X)$: $\{(1), (3)\}$.

How can we prove that $\Pi(D)$ contains a given atom ?

By exhibiting proofs for this atom.

Proof that $\overline{\mathbf{O}}(3)$ is contained in $\Pi(D)$

$$\overline{\mathbf{O}}(3), \mathbf{O}(3) \Rightarrow \overline{\mathbf{O}}(3)$$
$$\downarrow$$
$$\mathbf{O}(3), \mathbf{E}(2) \wedge \mathbf{S}(2,3) \Rightarrow \mathbf{O}(3)$$
$$\downarrow$$
$$\mathbf{E}(2), \mathbf{O}(1) \wedge \mathbf{S}(1,2) \Rightarrow \mathbf{E}(2)$$
$$\downarrow$$
$$\mathbf{O}(1), \mathbf{E}(0) \wedge \mathbf{S}(0,1) \Rightarrow \mathbf{O}(1)$$
$$\downarrow$$
$$\mathbf{E}(0), \mathbf{Z}(0) \Rightarrow \mathbf{E}(0)$$

How can we prove that $\Pi(D)$ contains a given atom ?

By exhibiting proofs for this atom.

Proof that $\overline{\mathbf{O}}(3)$ is contained in $\Pi(D)$

$$\overline{\mathbf{O}}(3), \mathbf{O}(3) \Rightarrow \overline{\mathbf{O}}(3)$$
$$\downarrow$$
$$\mathbf{O}(3), \mathbf{E}(X) \wedge \mathbf{S}(X, 3) \Rightarrow \mathbf{O}(3)$$
$$\downarrow$$
$$\mathbf{E}(X), \mathbf{O}(Y) \wedge \mathbf{S}(Y, X) \Rightarrow \mathbf{E}(X)$$
$$\downarrow$$
$$\mathbf{O}(Y), \mathbf{E}(W) \wedge \mathbf{S}(W, Y) \Rightarrow \mathbf{O}(Y)$$
$$\downarrow$$
$$\mathbf{E}(W), \mathbf{Z}(W) \Rightarrow \mathbf{E}(W)$$

Problem

We may use an unbounded number of variables.

- The proof of $\overline{\mathbf{O}}(2n+1)$ involves $2n+1$ variables.

Solution

We may re-use **forgotten** variables.

Proof that $\overline{\mathsf{O}}(3)$ is contained in $\Pi(D)$

$$\overline{\mathsf{O}}(3), \mathsf{O}(3) \Rightarrow \overline{\mathsf{O}}(3)$$
$$\downarrow$$
$$\mathsf{O}(3), \mathsf{E}(2) \wedge \mathsf{S}(2,3) \Rightarrow \mathsf{O}(3)$$
$$\downarrow$$
$$\mathsf{E}(2), \mathsf{O}(1) \wedge \mathsf{S}(1,2) \Rightarrow \mathsf{E}(2)$$
$$\downarrow$$
$$\mathsf{O}(1), \mathsf{E}(0) \wedge \mathsf{S}(0,1) \Rightarrow \mathsf{O}(1)$$
$$\downarrow$$
$$\mathsf{E}(0), \mathsf{Z}(0) \Rightarrow \mathsf{E}(0)$$

Proof that $\overline{\mathbf{O}}(3)$ is contained in $\Pi(D)$

$$\overline{\mathbf{O}}(3), \mathbf{O}(3) \Rightarrow \overline{\mathbf{O}}(3)$$

$$\downarrow$$

$$\mathbf{O}(3), \mathbf{E}(X) \wedge \mathbf{S}(X, 3) \Rightarrow \mathbf{O}(3)$$

$$\downarrow$$

$$\mathbf{E}(X), \mathbf{O}(Y) \wedge \mathbf{S}(Y, X) \Rightarrow \mathbf{E}(X)$$

$$\downarrow$$

$$\mathbf{O}(Y), \mathbf{E}(W) \wedge \mathbf{S}(W, Y) \Rightarrow \mathbf{O}(Y)$$

$$\downarrow$$

$$\mathbf{E}(W), \mathbf{Z}(W) \Rightarrow \mathbf{E}(W)$$

Proof that $\overline{\mathsf{O}}(3)$ is contained in $\Pi(D)$

$$\overline{\mathsf{O}}(3), \mathsf{O}(3) \Rightarrow \overline{\mathsf{O}}(3)$$
$$\downarrow$$
$$\mathsf{O}(3), \mathsf{E}(X) \wedge \mathsf{S}(X, 3) \Rightarrow \mathsf{O}(3)$$
$$\downarrow$$
$$\mathsf{E}(X), \mathsf{O}(Y) \wedge \mathsf{S}(Y, X) \Rightarrow \mathsf{E}(X)$$
$$\downarrow$$
$$\mathsf{O}(Y), \mathsf{E}(X) \wedge \mathsf{S}(X, Y) \Rightarrow \mathsf{O}(Y)$$
$$\downarrow$$
$$\mathsf{E}(X), \mathsf{Z}(X) \Rightarrow \mathsf{E}(X)$$

When are two Datalog programs A and B equivalent ?

- When A is contained in B :
  - When all the solutions of $\alpha(\mathbf{v})$ in $A(D)$ are solutions of $\beta(\mathbf{v})$ in $B(D)$, for every finite database D.
- When B is contained in A.

How can we check that A is contained in B ?

- For every proof of an atom $\alpha(\mathbf{v})$, we must find a proof of $\beta(\mathbf{v})$.
- We must find **containment mappings**.

Containment : Example

Program A

- $\mathsf{Z}(X) \Rightarrow \mathsf{E}(X)$
- $\mathsf{E}(X) \wedge \mathsf{S}(X, Y) \Rightarrow \mathsf{O}(Y)$
- $\mathsf{O}(X) \wedge \mathsf{S}(X, Y) \Rightarrow \mathsf{E}(Y)$
- $\mathsf{O}(X) \Rightarrow \overline{\mathsf{O}}(X)$
- Goal predicate : $\overline{\mathsf{O}}$.

Program B

- $\mathsf{Z}(X) \Rightarrow \mathsf{I}(X)$
- $\mathsf{I}(X) \wedge \mathsf{S}(X, Y) \Rightarrow \mathsf{I}(Y)$
- $\mathsf{I}(X) \Rightarrow \bar{\mathsf{I}}(X)$
- Goal predicate : $\bar{\mathsf{I}}$.

Containment relations

- A is contained in B.
- B is not contained in A.

Unfolding tree of A

$$\overline{\mathsf{O}}(X), \mathsf{O}(X) \Rightarrow \overline{\mathsf{O}}(X)$$
$$\downarrow$$
$$\mathsf{O}(X), \mathsf{E}(Y) \wedge \mathsf{S}(Y, X) \Rightarrow \mathsf{O}(X)$$
$$\downarrow$$
$$\mathsf{E}(Y), \mathsf{O}(V) \wedge \mathsf{S}(V, Y) \Rightarrow \mathsf{E}(Y)$$
$$\downarrow$$
$$\mathsf{O}(V), \mathsf{E}(W) \wedge \mathsf{S}(W, V) \Rightarrow \mathsf{O}(V)$$
$$\downarrow$$
$$\mathsf{E}(W), \mathsf{Z}(W) \Rightarrow \mathsf{E}(W)$$

Unfolding tree of B

$$\overline{\mathsf{I}}(\overline{X}), \mathsf{I}(\overline{X}) \Rightarrow \overline{\mathsf{I}}(\overline{X})$$
$$\downarrow$$
$$\mathsf{I}(\overline{X}), \mathsf{I}(\overline{Y}) \wedge \mathsf{S}(\overline{Y}, \overline{X}) \Rightarrow \mathsf{I}(\overline{X})$$
$$\downarrow$$
$$\mathsf{I}(\overline{Y}), \mathsf{I}(\overline{V}) \wedge \mathsf{S}(\overline{V}, \overline{Y}) \Rightarrow \mathsf{I}(\overline{Y})$$
$$\downarrow$$
$$\mathsf{I}(\overline{V}), \mathsf{I}(\overline{W}) \wedge \mathsf{S}(\overline{W}, \overline{V}) \Rightarrow \mathsf{I}(\overline{V})$$
$$\downarrow$$
$$\mathsf{I}(\overline{W}), \mathsf{Z}(\overline{W}) \Rightarrow \mathsf{I}(\overline{W})$$

How can we work on trees ?

- Words are recognized by **word** automata.
- Trees are recognized by **tree** automata.

Different tree automata

- Top-down deterministic finite automata
- **Top-down non-deterministic finite automata**
- Bottom-up deterministic finite automata
- Bottom-up non-deterministic finite automata
- Two-way alternating finite automata

$$\mathbf{TDDFA} \subsetneq \mathbf{TDNDFA} = \mathbf{BUDFA} = \mathbf{BDNDFA} = \mathbf{TWAFA}$$

What is a monadic program ?

A program in which each internal IDB predicate is of arity 1.

Example of monadic program

Program B

- $\mathbf{Z}(X) \Rightarrow \mathbf{I}(X)$
- $\mathbf{I}(X) \wedge \mathbf{S}(X, Y) \Rightarrow \mathbf{I}(Y)$
- $\mathbf{I}(X) \Rightarrow \bar{\mathbf{I}}(X)$
- Goal predicate : $\bar{\mathbf{I}}$

Decoration of a tree of A

- Identify the internal IDB predicates of B.
- Assume that some internal IDB predicates hold on some variables of the tree.
- Store the information
  - **globally** in unfolding trees.
  - **locally** in proof trees.

Example of decorated unfolding tree

- $\overline{\mathbf{O}}(X), \mathbf{O}(X) \Rightarrow \overline{\mathbf{O}}(X)$

  $\downarrow$

  $\mathbf{O}(X), \mathbf{E}(Y) \wedge \mathbf{S}(Y, X) \Rightarrow \mathbf{O}(X)$

  $\downarrow$

  $\mathbf{E}(Y), \mathbf{O}(V) \wedge \mathbf{S}(V, Y) \Rightarrow \mathbf{E}(Y)$

  $\downarrow$

  $\mathbf{O}(V), \mathbf{E}(W) \wedge \mathbf{S}(W, V) \Rightarrow \mathbf{O}(V)$

  $\downarrow$

  $\mathbf{E}(W), \mathbf{Z}(W) \Rightarrow \mathbf{E}(W)$

Example of decorated unfolding tree

- $\overline{\mathbf{O}}(X), \mathbf{O}(X) \Rightarrow \overline{\mathbf{O}}(X)$

  $\downarrow$

  $\mathbf{O}(X), \mathbf{E}(Y) \wedge \mathbf{S}(Y, X) \Rightarrow \mathbf{O}(X)$

  $\downarrow$

  $\mathbf{E}(Y), \mathbf{O}(V) \wedge \mathbf{S}(V, Y) \Rightarrow \mathbf{E}(Y)$

  $\downarrow$

  $\mathbf{O}(V), \mathbf{E}(W) \wedge \mathbf{S}(W, V) \Rightarrow \mathbf{O}(V)$

  $\downarrow$

  $\mathbf{E}(W), \mathbf{Z}(W) \Rightarrow \mathbf{E}(W)$
- IDB predicates : $\mathbf{I}$.
- Assumed relations : $\mathbf{I}(X), \mathbf{I}(Y), \mathbf{I}(V), \mathbf{I}(W)$.

Example of decorated proof tree

$$\overline{\mathsf{O}}(X), \mathsf{O}(X) \Rightarrow \overline{\mathsf{O}}(X)$$
$$\downarrow$$
$$\mathsf{O}(X), \mathsf{E}(Y) \wedge \mathsf{S}(Y, X) \Rightarrow \mathsf{O}(X)$$
$$\downarrow$$
$$\mathsf{E}(Y), \mathsf{O}(X) \wedge \mathsf{S}(X, Y) \Rightarrow \mathsf{E}(Y)$$
$$\downarrow$$
$$\mathsf{O}(X), \mathsf{E}(Y) \wedge \mathsf{S}(Y, X) \Rightarrow \mathsf{O}(X)$$
$$\downarrow$$
$$\mathsf{E}(Y), \mathsf{Z}(Y) \Rightarrow \mathsf{E}(Y)$$

Example of decorated proof tree

$$\overline{\mathsf{O}}(X), \mathsf{O}(X) \Rightarrow \overline{\mathsf{O}}(X) \qquad \text{— I} \quad \text{— I(X)}$$

$$\downarrow$$

$$\mathsf{O}(X), \mathsf{E}(Y) \wedge \mathsf{S}(Y, X) \Rightarrow \mathsf{O}(X) \qquad \text{— I} \quad \text{— I(X), I(Y)}$$

$$\downarrow$$

$$\mathsf{E}(Y), \mathsf{O}(X) \wedge \mathsf{S}(X, Y) \Rightarrow \mathsf{E}(Y) \qquad \text{— I} \quad \text{— I(X), I(Y)}$$

$$\downarrow$$

$$\mathsf{O}(X), \mathsf{E}(Y) \wedge \mathsf{S}(Y, X) \Rightarrow \mathsf{O}(X) \qquad \text{— I} \quad \text{— I(X), I(Y)}$$

$$\downarrow$$

$$\mathsf{E}(Y), \mathsf{Z}(Y) \Rightarrow \mathsf{E}(Y) \qquad \text{— I} \quad \text{— I(Y)}$$

What is a fix-point tree ?

- Nothing more is implied by the assumptions.
- There exists a **least** fix-point.

Why are these trees important ?

Equivalence between :

- A proof of $\mathbf{P}(X)$.
- The presence of $\mathbf{P}(X)$ in the least fix-point.
- The presence of $\mathbf{P}(X)$ in **every** fix-point.

How can we recognize fix-point proof trees ?

With automata :

- We search an infix-point certificate.
- The research fails when the tree is fix-point.

How can we conclude over the containment ?

With automata :

- We search a reaching-goal certificate.
- The research succeeds when a containment mapping exists.

What is a certificate ?

A mapping of variables :

- From the variables of a rule.
- To the variables of a tree.
- That stabilizes the body predicates.
- That does not stabilize the head predicate.

How do we search certificates ?

With top-down non-deterministic automata :

- We identify mapped variables in the current node.
- Unidentified variables are in one child sub-tree.
- We look for stabilized body predicates in the current node.
- We check if the head predicate of the rule was stabilized.

Research of an infix-point certificate : Example

- IDB predicates : $\mathbf{E}, \mathbf{O}$.
- Rule : $\mathbf{E}(\overline{X}) \wedge \mathbf{T}(\overline{Y}, \overline{Z}) \wedge \mathbf{T}(\overline{X}, \overline{Y}) \Rightarrow \mathbf{O}(\overline{Y})$.
- Current sub-tree :

$$\mathbf{G}(Y), \mathbf{U}(Y) \wedge \mathbf{H}(Y) \Rightarrow \mathbf{G}(Y) \text{ — } \mathbf{E}, \mathbf{O} \text{ — } \mathbf{E}(Y)$$

$$\downarrow \qquad\qquad\qquad\qquad \downarrow$$

$\mathbf{U}(Y)$,
$\mathbf{T}(Y, Z) \wedge \mathbf{T}(Z, W) \Rightarrow \mathbf{U}(Y)$
$\mathbf{E}, \mathbf{O}$ — $\mathbf{E}(Y)$

$\mathbf{H}(Y)$,
$\mathbf{T}(X, Y) \Rightarrow \mathbf{H}(Y)$
$\mathbf{E}, \mathbf{O}$ — $\mathbf{E}(Y), \mathbf{E}(X)$

Research of an infix-point certificate : Example

- IDB predicates : $\mathbf{E}, \mathbf{O}$.
- Rule : $\mathbf{E}(\overline{X}) \wedge \mathbf{T}(\overline{Y}, \overline{Z}) \wedge \mathbf{T}(\overline{X}, \overline{Y}) \Rightarrow \mathbf{O}(\overline{Y})$.
- Current sub-tree :

$$\mathbf{G}(Y), \mathbf{U}(Y) \wedge \mathbf{H}(Y) \Rightarrow \mathbf{G}(Y) \text{ — } \mathbf{E}, \mathbf{O} \text{ — } \mathbf{E}(Y), \mathbf{O}(Y)$$

$$\downarrow \qquad\qquad\qquad\qquad \downarrow$$

$\mathbf{U}(Y)$,
$\mathbf{T}(Y, Z) \wedge \mathbf{T}(Z, W) \Rightarrow \mathbf{U}(Y)$
$\mathbf{E}, \mathbf{O}$ — $\mathbf{E}(Y), \mathbf{O}(Y)$

$\mathbf{H}(Y)$,
$\mathbf{T}(X, Y) \Rightarrow \mathbf{H}(Y)$
$\mathbf{E}, \mathbf{O}$ — $\mathbf{E}(Y), \mathbf{E}(X), \mathbf{O}(Y)$

We build automata checking whether the tree

- Is a decorated prof tree.
- Contains an infix-point certificate.
- Contains a reaching-goal certificate.

A is contained in B when every decorated proof tree of A

- Contains an infix-point certificate.
- Contains a reaching-goal certificate.

We proceed by

- Union of automata.
- Complementation of automata.
- Intersection of automata.
- Emptiness-checking of automata.

Complexity of the algorithm

2EXPTIME in the sizes of A and B

What is a transitive program ?

A program with recursion through transitive closure only.

Example of transitive program

Program C

- $\top \Rightarrow \mathbf{S}^*(X, X)$
- $\mathbf{S}(X, Y) \wedge \mathbf{S}^*(Y, Z) \Rightarrow \mathbf{S}^*(X, Z)$
- $\mathbf{Z}(X) \wedge \mathbf{S}^*(X, Y) \Rightarrow \mathbf{I}(Y)$
- Goal predicate : $\mathbf{I}$

Diamond reduction

Non-recursive program with

- New EDB **diamond** predicates.
- New **diamond** rules.
- Rules $\top \Rightarrow \mathbf{S}^*(X, X)$ are erased.
- Rules $\mathbf{S}(X, Y) \wedge \mathbf{S}^*(Y, Z) \Rightarrow \mathbf{S}^*(X, Z)$ are replaced by diamond rules.

Twelve diamond rules

1. $\mathsf{S}(X, X_1) \wedge \mathsf{S}^\diamond(X_1, X_2) \wedge \mathsf{S}(X_2, X_3) \wedge \mathsf{S}^\diamond(X_3, X_4) \wedge \mathsf{S}(X_4, Y) \Rightarrow \mathsf{S}^*(X, Y)$
2. $\mathsf{S}(X, X_1) \wedge \mathsf{S}^\diamond(X_1, X_2) \wedge \mathsf{S}(X_2, X_3) \wedge \mathsf{S}^\diamond(X_3, Y) \Rightarrow \mathsf{S}^*(X, Y)$
3. $\mathsf{S}(X, X_1) \wedge \mathsf{S}^\diamond(X_1, X_2) \wedge \mathsf{S}^\diamond(X_2, X_3) \wedge \mathsf{S}(X_3, Y) \Rightarrow \mathsf{S}^*(X, Y)$
4. $\mathsf{S}(X, X_1) \wedge \mathsf{S}^\diamond(X_1, X_2) \wedge \mathsf{S}^\diamond(X_2, Y) \Rightarrow \mathsf{S}^*(X, Y)$
5. $\mathsf{S}^\diamond(X, X_1) \wedge \mathsf{S}(X_1, X_2) \wedge \mathsf{S}^\diamond(X_2, X_3) \wedge \mathsf{S}(X_3, Y) \Rightarrow \mathsf{S}^*(X, Y)$
6. $\mathsf{S}^\diamond(X, X_1) \wedge \mathsf{S}(X_1, X_2) \wedge \mathsf{S}^\diamond(X_2, Y) \Rightarrow \mathsf{S}^*(X, Y)$
7. $\mathsf{S}^\diamond(X, X_1) \wedge \mathsf{S}^\diamond(X_1, X_2) \wedge \mathsf{S}(X_2, Y) \Rightarrow \mathsf{S}^*(X, Y)$
8. $\mathsf{S}^\diamond(X, X_1) \wedge \mathsf{S}^\diamond(X_1, Y) \Rightarrow \mathsf{S}^*(X, Y)$
9. $\mathsf{S}(X, X_1) \wedge \mathsf{S}^\diamond(X_1, X_2) \wedge \mathsf{S}(X_2, Y) \Rightarrow \mathsf{S}^*(X, Y)$
10. $\mathsf{S}(X, X_1) \wedge \mathsf{S}^\diamond(X_1, Y) \Rightarrow \mathsf{S}^*(X, Y)$
11. $\mathsf{S}^\diamond(X, X_1) \wedge \mathsf{S}(X_1, Y) \Rightarrow \mathsf{S}^*(X, Y)$
12. $\mathsf{S}^\diamond(X, Y) \Rightarrow \mathsf{S}^*(X, Y)$

Labelling of a proof tree

- Select atoms of an unfolding tree of the diamond reduction.
- Map variables of these atoms to variables of the current node.
- Build a logical formula from this set and this mapping.
- Store the couple (Set of atoms, Associated mapping) **locally**.
- Store **locally** a set a such couples.

Example of formula

- Current node : $\mathbf{I}(X), \mathbf{S}(Y, X) \wedge \mathbf{I}(Y) \Rightarrow \mathbf{I}(X)$
- Set of atoms : $\{\mathbf{S}^{\diamond}(\overline{X}, \overline{Y}), \mathbf{S}(\overline{Y}, \overline{Z})\}$
- Mapping : $\overline{X} \to X, \overline{Y} \to Y, \overline{Z} \to \overline{Z}$
- Logical formula : $\left(\exists \overline{Z}\right) \left(\mathbf{S}^{*}(X, Y) \wedge \mathbf{S}(Y, \overline{Z})\right)$

Example of labelled proof tree

$\bar{\mathsf{I}}(X),$
$\mathsf{I}(X) \Rightarrow \bar{\mathsf{I}}(X)$

$\downarrow$

$\mathsf{I}(X),$
$\mathsf{I}(Y) \wedge \mathsf{S}(Y,X) \Rightarrow \mathsf{I}(X)$

$\downarrow$

$\mathsf{I}(Y),$
$\mathsf{Z}(Y) \Rightarrow \mathsf{I}(Y)$

Example of labelled proof tree

$$\bar{\mathsf{I}}(X), \qquad \{(\{\mathsf{S}^\diamond(\overline{X},\overline{Y}), \mathsf{S}(\overline{Y},\overline{Z})\},$$
$$\mathsf{I}(X) \Rightarrow \bar{\mathsf{I}}(X) \qquad \{\overline{X} \to X, \overline{Y} \to \overline{Y}, \overline{Z} \to \overline{Z}\})\}$$

$$\downarrow$$

$$\mathsf{I}(X), \qquad \{(\{\mathsf{S}^\diamond(\overline{X},\overline{Y}), \mathsf{S}(\overline{Y},\overline{Z})\},$$
$$\mathsf{I}(Y) \wedge \mathsf{S}(Y,X) \Rightarrow \mathsf{I}(X) \qquad \{\overline{X} \to X, \overline{Y} \to Y, \overline{Z} \to \overline{Z}\})\}$$

$$\downarrow$$

$$\mathsf{I}(Y), \qquad \{(\{\mathsf{S}^\diamond(\overline{X},\overline{Y}), \mathsf{S}(\overline{Y},\overline{Z})\},$$
$$\mathsf{Z}(Y) \Rightarrow \mathsf{I}(Y) \qquad \{\overline{X} \to \overline{X}, \overline{Y} \to Y, \overline{Z} \to \overline{Z}\})\}$$

What is a fix-point labelled proof tree ?

- The stored formulæ do not imply **directly** other formulæ.
- There exists a **least** fix-point.

Why are these trees important ?

Equivalence between :

- A proof of $\beta(\mathbf{v})$.
- The presence of $\beta(\mathbf{v})$ in the root of the least fix-point.
- The presence of $\beta(\mathbf{v})$ in the root of **every** fix-point.

What is a **direct** implication ?

1. If φ is true in a neighbour node, φ is true.
2. If $\varphi_1 \wedge \varphi_2$ is true, φ_1 is true.
3. If φ_1 and φ_2 are true, $\varphi_1 \wedge \varphi_2$ is true.
4. If φ is true, $(\exists \mathbf{v})(\varphi)$ is true.
5. If $(\exists \mathbf{v})(\varphi_1 \wedge \varphi_2)$ is true and $\varphi_2 \Rightarrow \varphi_3$ is a rule, $(\exists \mathbf{v})(\varphi_1 \wedge \varphi_2 \wedge \varphi_3)$ is true.
6. If φ is an EDB atom of the current node, φ is true.
7. If $(\exists \mathbf{v})(\varphi)$ is true, $(\exists \mathbf{v})(\varphi \wedge \mathbf{S}^*(x, x))$ is true.
8. If $(\exists \mathbf{v})(\varphi \wedge \mathbf{S}(x, y))$ is true, $(\exists \mathbf{v})(\varphi \wedge \mathbf{S}^*(x, y))$ is true.
9. If $(\exists \mathbf{v})(\varphi \wedge \mathbf{S}^*(x, y) \wedge \mathbf{S}^*(y, z))$ is true, $(\exists \mathbf{v})(\varphi \wedge \mathbf{S}^*(x, z))$ is true.
10. $\top$ is true.

How do we recognize fix-point labelled proof trees ?

With top-down non-deterministic automata :

- We select a direct implication rule.
- We select one or two true formulæ stored in the node.
- We compute the implied formula.
- We verify that this formula is stored in the node.

How can we conclude over the containment ?

With automata :

- The goal formula $\beta(\mathbf{v})$ is stored in the root when a containment mapping exists.

We build an automaton checking whether the tree

- Is a fix-point labelled prof tree.
- Does not store the goal formula $\beta(\mathbf{v})$ in its root.

A is contained in B when every labelled proof tree of A

- Stores the goal formula $\beta(\mathbf{v})$ in its root.

We proceed by

- Emptiness-checking of automata.

Complexity of the algorithm

3EXPTIME in the sizes of A and B

We already knew

- The decidability of the containment problem in monadic and transitive programs.
- Algorithms of non-elementary complexity.

I found

- An algorithm deciding the containment in monadic programs in 2EXPTIME.
- An algorithm deciding the containment in transitive programs in 3EXPTIME.

We still have to

- Find lower bounds for the containment problems.
- Search algorithms deciding directly the equivalence problems.

Questions ?