

# A Logic-based Framework for Reasoning about Composite Data Structures

Ahmed Bouajjani, Cezara Drăgoi, Constantin Enea, and Mihaela Sighireanu

LIAFA , CNRS & University Paris 7, France  
175, rue du Chevaleret, 75013 Paris, France  
E-mail: [cezara.dragoi@liafa.jussieu.fr](mailto:cezara.dragoi@liafa.jussieu.fr)

# Program verification

The verification of:

- sequential programs
- manipulating dynamic data structures, like
  - array,
  - singly-linked lists,
  - doubly-linked lists,
  - array of doubly-linked lists
- strongly typed
- allowed types:  $\left\{ \begin{array}{l} - \text{basic types: int, bool, etc.} \\ - \text{references to records} \\ - \text{array of records} \end{array} \right.$

The data structures have **arbitrary size and shape** and encapsulate **data over unbounded domains**.

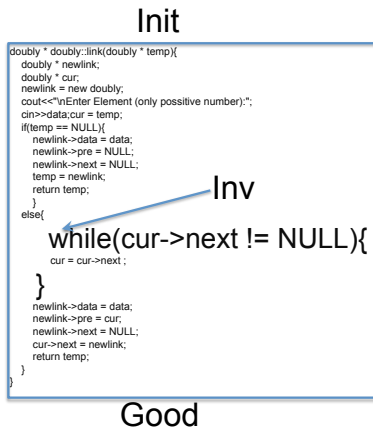


Verification approach:

- pre/post condition reasoning
- invariants checking

Check that:

- $post(Inv) \subseteq Inv$
- $Inv \subseteq Good$



# Our contribution

We introduce a logic called **C**omposite **S**tructures **L**ogic that:

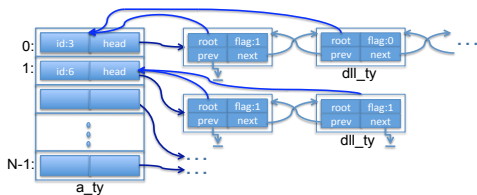
- is able to express **properties on the shape** of the heap as well as **properties on the size** of the heap as well as **properties on the data** stored in the heap
- can specify the effect of executing transformations on these data structures, it is **closed under post-computation**
- has a **decidable satisfiability problem**

- 1 General logic: gCSL
- 2 CSL
- 3 Decision procedure for CSL
- 4 Related work
- 5 Conclusions and future work

# Heaps as graphs

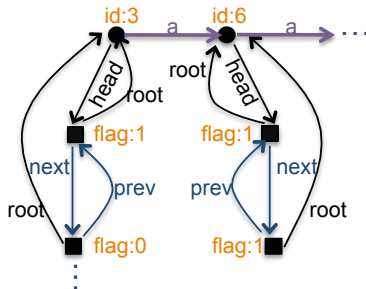
Heaps are represented as labeled directed graphs called **heap graphs**

```
struct a_ty {  
    int id;  
    dll_ty* head;  
}  
  
struct dll_ty {  
    bool flag;  
    a_ty* root;  
    dll_ty* next, *prev;  
}  
  
a_ty arr[N];
```



The graph is **deterministic**

The array fields create **acyclic distinct paths**



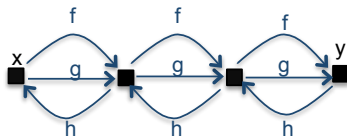
- assume  $\mathbb{D}$  the data domain where data fields take values
- assume  $\text{FO}(\mathbb{D}, \mathbb{O}, \mathbb{P})$  a first order logic on  $\mathbb{D}$ , with operations in  $\mathbb{O}$  and predicates in  $\mathbb{P}$

gCSL is a **multi-sorted first order logic on graphs** parametrized by  $\text{FO}(\mathbb{D}, \mathbb{O}, \mathbb{P})$

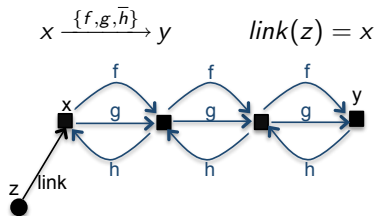
$$\text{gCSL} = \text{FO} + \text{reachability} + \begin{array}{l} \text{arithmetical} \\ \text{constraints} \end{array} + \text{FO}(\mathbb{D}, \mathbb{O}, \mathbb{P})$$

# Reachability predicates in gCSL

$$x \xrightarrow{\{f, g, \bar{h}\}} y$$



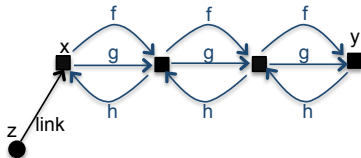
# Reachability predicates in gCSL



# Reachability predicates and length constraints in gCSL

$$x \xrightarrow{\{f, g, \bar{h}\}, !} y$$

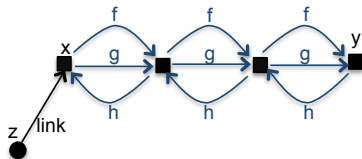
$$x \xrightarrow{\{f, g, \bar{h}\}} y \quad \text{link}(z) = x$$



# Reachability predicates and length constraints in gCSL

$$x \xrightarrow{\{f, g, \bar{h}\}, l} y \wedge l = 3$$

$$x \xrightarrow{\{f, g, \bar{h}\}} y \quad \text{link}(z) = x$$

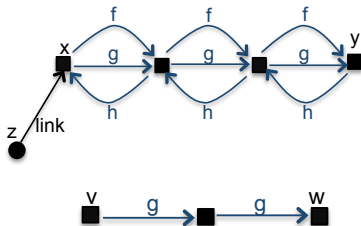


# Reachability predicates and length constraints in gCSL

$$x \xrightarrow{\{f, g, \bar{h}\}, l} y \wedge l = 3 \wedge v \xrightarrow{\{g\}, l'} w$$

$$l' < l \wedge l + l' \geq 4$$

$$x \xrightarrow{\{f, g, \bar{h}\}} y \quad \text{link}(z) = x$$



# Data constraints in gCSL

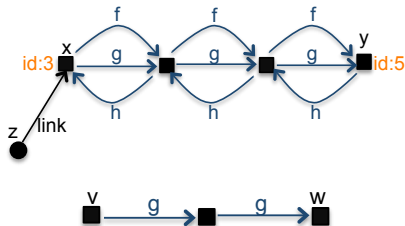
$$id(x) = 3$$

$$\exists c. id(x) + id(y) + c \geq 9$$

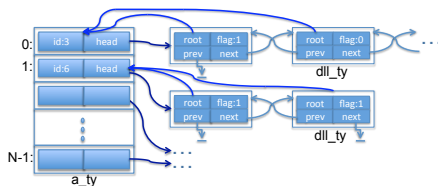
$$x \xrightarrow{\{f, g, \bar{h}\}, l} y \wedge l = 3 \wedge v \xrightarrow{\{g\}, l'} w$$

$$l' < l \wedge l + l' \geq 4$$

$$x \xrightarrow{\{f, g, \bar{h}\}} y \quad link(z) = x$$



# Program verification



**Structure:** “the array contains in each cell a reference to an acyclic doubly linked list”

$$\forall i \exists x, y. x = \text{head}(a[i]) \wedge x \xrightarrow{\{\text{next}, \overline{\text{prev}}\}} y$$

**Sizes:** “the array is sorted w.r.t. the lengths of lists stored”

$$\forall j, j'. j < j' \implies \exists x, x', l, l'. (x = \text{head}(a[j]) \wedge x' = \text{head}(a[j']) \wedge x \xrightarrow{\{\text{next}\}, l} \text{null} \wedge x' \xrightarrow{\{\text{next}\}, l'} \text{null} \wedge l' \leq l)$$

**Data:** “the array is sorted w.r.t. the values of the field id”

$$\forall i, j. i < j \implies \text{id}(a[i]) < \text{id}(a[j])$$

The satisfiability problem of gCSL is **undecidable**

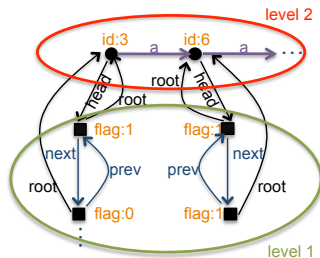
- when data are restricted to finite domains (such as booleans), the logic subsumes **the first-order logic on graphs with reachability**
- when the models are restricted to simple structures, like sequences or arrays, for very simple data logics such as  $(\mathbb{N}, =)$ , **the fragment  $\forall^*\exists^*$**  is undecidable

- 1 General logic: gCSL
- 2 CSL**
- 3 Decision procedure for CSL
- 4 Related work
- 5 Conclusions and future work

# CSL Logic

An **ordered partition** over  $\mathcal{RT}$  is a mapping  $\sigma : \mathcal{RT} \rightarrow \{1, \dots, N\}$

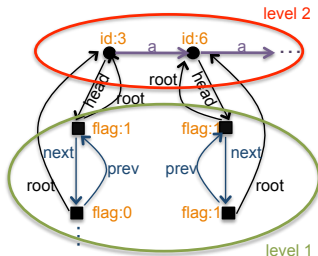
- a type  $R \in \mathcal{RT}$  is of **level  $k$**  iff  $\sigma(R) = k$



# CSL Logic

An **ordered partition** over  $\mathcal{RT}$  is a mapping  $\sigma : \mathcal{RT} \rightarrow \{1, \dots, N\}$

- a type  $R \in \mathcal{RT}$  is of **level**  $k$  iff  $\sigma(R) = k$



For  $1 \leq k \leq |\sigma|$ ,

**CSL** is the smallest set of formulas closed under disjunction and conjunction, which contains all the closed formulas of the form:

$$\exists_{\leq k}^* \forall_k^* \exists_{\leq k-1}^* \forall_{k-1}^* \dots \exists_1^* \forall_1^* \{\exists_d, \forall_d\}^* . \phi$$

$\phi$  is a quantifier-free formula in gCSL

# CSL Logic

For  $1 \leq k \leq |\sigma|$ ,

CSL is the smallest set of formulas closed under disjunction and conjunction, which contains all the closed formulas of the form:

$$\exists_{\leq k}^* \forall_k^* \exists_{\leq k-1}^* \forall_{k-1}^* \dots \exists_1^* \forall_1^* \{\exists_d, \forall_d\}^* . \phi$$

$\phi$  is a quantifier-free formula in gCSL such that:

# CSL Logic

For  $1 \leq k \leq |\sigma|$ ,

**CSL** is the smallest set of formulas closed under disjunction and conjunction, which contains all the closed formulas of the form:

$$\exists_{\leq k}^* \forall_k^* \exists_{\leq k-1}^* \forall_{k-1}^* \dots \exists_1^* \forall_1^* \{\exists_d, \forall_d\}^* . \phi$$

$\phi$  is a quantifier-free formula in gCSL such that:

- **REACH**: for any  $x \xrightarrow{A, ind} x'$ ,  $x$  and  $x'$  are free or existential variables

$$\text{YES} \quad \exists x, x'. x \xrightarrow{\{f\}, l} x'$$

$$\text{NO} \quad \forall y, y'. y \xrightarrow{\{f\}, l} y'$$

# CSL Logic

For  $1 \leq k \leq |\sigma|$ ,

CSL is the smallest set of formulas closed under disjunction and conjunction, which contains all the closed formulas of the form:

$$\exists_{\leq k}^* \forall_k^* \exists_{\leq k-1}^* \forall_{k-1}^* \dots \exists_1^* \forall_1^* \{\exists_d, \forall_d\}^* . \phi$$

$\phi$  is a quantifier-free formula in gCSL such that:

- **REACH**: for any  $x \xrightarrow{A, ind} x'$ ,  $x$  and  $x'$  are free or existential variables
- **UNIVIDX**: two universal index variables can be used only in  $j < j'$  or  $j = j'$

$$\text{YES} \quad \forall j, j'. j < j' \Rightarrow \text{data}(a[j]) < \text{data}(a[j'])$$

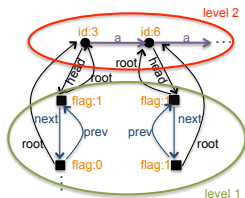
$$\text{NO} \quad \forall j, j'. j + 1 = j' \Rightarrow \text{data}(a[j]) < \text{data}(a[j'])$$

- **LEV**: atomic constraints on lengths of lists and array indexes involve only one level

$$\text{YES} \quad \exists x, x', h_1 \exists z, z', h_2. x \xrightarrow{\{f\}, h_1} x' \wedge z \xrightarrow{\{f\}, h_2} z' \wedge h_1 \geq 4 \wedge h_2 \geq 0$$

$$\text{NO} \quad \exists x, x', h_1 \exists z, z', h_2. x \xrightarrow{\{f\}, h_1} x' \wedge z \xrightarrow{\{f\}, h_2} z' \wedge h_1 + h_2 \geq 0$$

# Examples of CSL formulas



**Structure:** “the array contains in each cell a reference to an acyclic doubly linked list”

$$\forall i \exists x, y. x = \text{head}(a[i]) \wedge x \xrightarrow{\{\text{next}, \overline{\text{prev}}\}} y$$

**Sizes:** “the array is sorted w.r.t. the lengths of lists stored”

$$\forall j, j'. j < j' \implies \exists x, x', l, l'. (x = \text{head}(a[j]) \wedge x' = \text{head}(a[j']) \wedge x \xrightarrow{\{\text{next}\}, l} \text{null} \wedge x' \xrightarrow{\{\text{next}\}, l'} \text{null} \wedge l' \leq l)$$

**Data:** “the array is sorted w.r.t. the values of the field id”

$$\forall i, j. i < j \implies \text{id}(a[i]) < \text{id}(a[j])$$

- 1 General logic: gCSL
- 2 CSL
- 3 Decision procedure for CSL**
- 4 Related work
- 5 Conclusions and future work

# Satisfiability of CSL

## Theorem

*The satisfiability of CSL is decidable if the satisfiability of the underlying first order logic  $\text{FO}(\mathbb{D}, \mathbb{O}, \mathbb{P})$  is decidable*

Let

$$\varphi_k = \exists_{\leq k}^* \mathbf{r} \forall_k^* \mathbf{p} \exists_{\leq k-1}^* \mathbf{r}' \forall_{k-1}^* \mathbf{p}' \dots \exists_1^* \mathbf{r}'' \forall_1^* \mathbf{p}'' \{\exists_d, \forall_d\}^* . \phi$$

- 1 compute  $\varphi_{k-1}$  equi-satisfiable to  $\varphi_k$  such that

$$\varphi_{k-1} = \exists_{\leq k-1}^* \mathbf{z} \forall_{k-1}^* \mathbf{w} \dots \exists_1^* \mathbf{z}' \forall_1^* \mathbf{w}' \{\exists_d, \forall_d\}^* . \phi'$$

until it ends up with a formula over variables of level 1

$$\varphi = \exists_1^* \mathbf{x} \forall_1^* \mathbf{y} \{\exists_d, \forall_d\}^* . \phi''$$

- 2 reduce the satisfiability of  $\varphi$  to the satisfiability of a formula in  $\text{FO}(\mathbb{D}, \mathbb{O}, \mathbb{P})$

## Theorem

*The satisfiability of CSL is decidable if the satisfiability of the underlying first order logic  $FO(\mathbb{D}, \mathbb{O}, \mathbb{P})$  is decidable*

Let

$$\varphi = \exists_1^* \mathbf{x} \forall_1^* \mathbf{y} \{ \exists_d, \forall_d \}^* . \phi''$$

- 1 compute the **set of small models** for the reachability and size constraints
- 2 for each small model, build a  $FO(\mathbb{D}, \mathbb{O}, \mathbb{P})$  formula  $\psi$

If one of  $\psi$  is satisfiable then  $\varphi$  is satisfiable.

# Decision procedure: computing small models

$$\varphi = \exists x, q, z . x \xrightarrow{\{f\}} q \wedge x \xrightarrow{\{f\}} z \wedge q \neq z$$

# Decision procedure: computing small models

$$\varphi = \exists x, q, z . x \xrightarrow{\{f\}} q \wedge x \xrightarrow{\{f\}} z \wedge q \neq z$$



# Decision procedure: computing small models

$$\varphi = \exists x, q, z . x \xrightarrow{\{f\}} q \wedge x \xrightarrow{\{f\}} z \wedge q \neq z$$

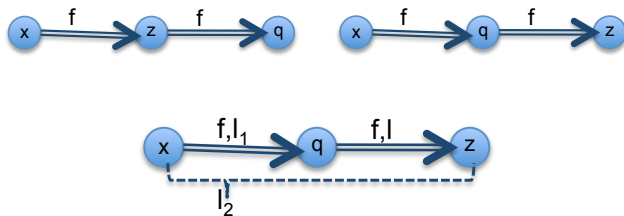


- $\varphi$  has two small models of size three



# Decision procedure: computing small models

$$\varphi = \exists x, q, z \exists l_1, l_2. x \xrightarrow{\{f\}, l_1} q \wedge x \xrightarrow{\{f\}, l_2} z \wedge q \neq z \\ \wedge l_1 + l_2 \geq 8$$



$$l_1 + l_2 = l_2 \wedge l_1 + l_2 \geq 8$$

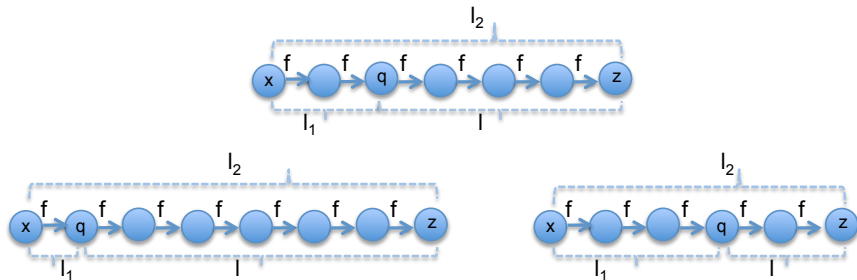
# Decision procedure: computing small models

$$\varphi = \exists x, q, z \exists l_1, l_2. x \xrightarrow{\{f\}, l_1} q \wedge x \xrightarrow{\{f\}, l_2} z \wedge q \neq z \\ \wedge l_1 + l_2 \geq 8$$

- minimal solutions  $(l_1, l_2, l)$  for  $l_1 + l = l_2 \wedge l_1 + l_2 \geq 8$

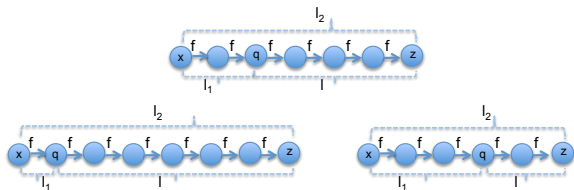
$$\mathcal{M} = \{(1, 7, 6), (2, 6, 4), (3, 5, 2)\}$$

- small models for  $\varphi$



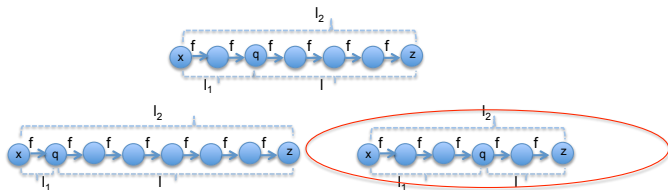
# Decision procedure: data constraints

$$\begin{aligned}\varphi = & \exists x, q, z \exists l_1, l_2. x \xrightarrow{\{f\}, l_1} q \wedge x \xrightarrow{\{f\}, l_2} z \wedge q \neq z \\ & \wedge l_1 + l_2 \geq 8 \\ & \wedge g(x) = 0 \wedge g(q) = 2 \\ & \wedge \forall y, y'. (y \xrightarrow{\{f\}} y' \Rightarrow g(y) < g(y'))\end{aligned}$$



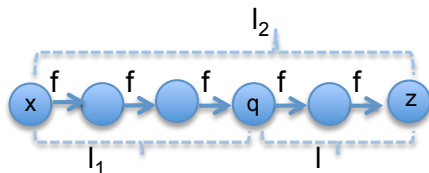
# Decision procedure: data constraints

$$\begin{aligned}\varphi = & \exists x, q, z \exists l_1, l_2. x \xrightarrow{\{f\}, l_1} q \wedge x \xrightarrow{\{f\}, l_2} z \wedge q \neq z \\ & \wedge l_1 + l_2 \geq 8 \\ & \wedge g(x) = 0 \wedge g(q) = 2 \\ & \wedge \forall y, y'. (y \xrightarrow{\{f\}} y' \Rightarrow g(y) < g(y'))\end{aligned}$$



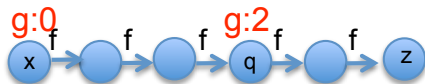
# Decision procedure: data constraints

$$\begin{aligned} \varphi = & \exists x, q, z \exists l_1, l_2. x \xrightarrow{\{f\}, l_1} q \wedge x \xrightarrow{\{f\}, l_2} z \wedge q \neq z \\ & \wedge l_1 + l_2 \geq 8 \\ & \wedge g(x) = 0 \wedge g(q) = 2 \\ & \wedge \forall y, y'. (y \xrightarrow{\{f\}} y' \Rightarrow g(y) < g(y')) \end{aligned}$$



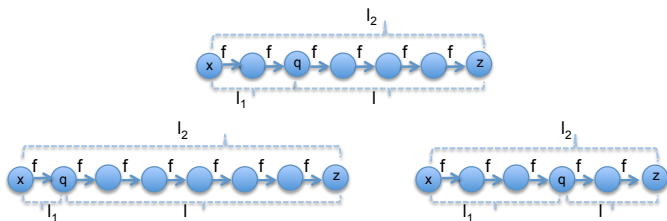
# Decision procedure: data constraints

$$\begin{aligned}\varphi = & \exists x, q, z \exists l_1, l_2. x \xrightarrow{\{f\}, l_1} q \wedge x \xrightarrow{\{f\}, l_2} z \wedge q \neq z \\ & \wedge l_1 + l_2 \geq 8 \\ & \wedge g(x) = 0 \wedge g(q) = 2 \\ & \wedge \forall y, y'. (y \xrightarrow{\{f\}} y' \Rightarrow g(y) < g(y'))\end{aligned}$$



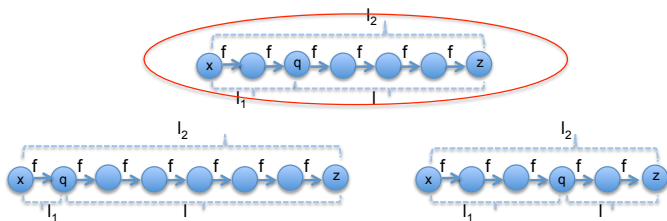
# Decision procedure: data constraints

$$\begin{aligned}\varphi = & \exists x, q, z \exists l_1, l_2. x \xrightarrow{\{f\}, l_1} q \wedge x \xrightarrow{\{f\}, l_2} z \wedge q \neq z \\ & \wedge l_1 + l_2 \geq 8 \\ & \wedge g(x) = 0 \wedge g(q) = 2 \\ & \wedge \forall y, y'. (y \xrightarrow{\{f\}} y' \Rightarrow g(y) < g(y'))\end{aligned}$$



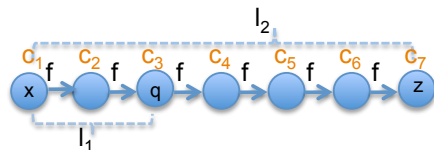
# Decision procedure: data constraints

$$\begin{aligned}\varphi = & \exists x, q, z \exists l_1, l_2. x \xrightarrow{\{f\}, l_1} q \wedge x \xrightarrow{\{f\}, l_2} z \wedge q \neq z \\ & \wedge l_1 + l_2 \geq 8 \\ & \wedge g(x) = 0 \wedge g(q) = 2 \\ & \wedge \forall y, y'. (y \xrightarrow{\{f\}} y' \Rightarrow g(y) < g(y'))\end{aligned}$$



# Decision procedure: data constraints

$$\begin{aligned}\varphi = & \exists x, q, z \exists l_1, l_2. x \xrightarrow{\{f\}, l_1} q \wedge x \xrightarrow{\{f\}, l_2} z \wedge q \neq z \\ & \wedge l_1 + l_2 \geq 8 \\ & \wedge g(x) = 0 \wedge g(q) = 2 \\ & \wedge \forall y, y'. (y \xrightarrow{\{f\}} y' \Rightarrow g(y) < g(y'))\end{aligned}$$



$$\begin{aligned}\psi_1 = & \exists c_1, c_2, c_3, c_4, c_5, c_5, c_6, c_7. \bigwedge_{i \neq j} c_i \neq c_j \\ & \text{true} \wedge \text{true} \wedge \text{true} \wedge \text{true} \\ & \wedge c_1 = 0 \wedge c_3 = 2 \\ & \wedge \bigwedge_{1 \leq i < j \leq 7} c_i < c_j\end{aligned}$$

# Decision procedure: data constraints

$$\begin{aligned}\varphi = & \exists x, q, z \exists l_1, l_2. x \xrightarrow{\{f\}, l_1} q \wedge x \xrightarrow{\{f\}, l_2} z \wedge q \neq z \\ & \wedge l_1 + l_2 \geq 8 \\ & \wedge g(x) = 0 \wedge g(q) = 2 \\ & \wedge \forall y, y'. (y \xrightarrow{\{f\}} y' \Rightarrow g(y) < g(y'))\end{aligned}$$



$$\begin{aligned}\psi_1 = & \exists c_1, c_2, c_3, c_4, c_5, c_5, c_6, c_7. \bigwedge_{i \neq j} c_i \neq c_j \\ & true \wedge true \wedge true \wedge true \\ & \wedge c_1 = 0 \wedge c_3 = 2 \\ & \wedge \bigwedge_{1 \leq i < j \leq 7} c_i < c_j\end{aligned}$$

# Decision procedure: summary

- 1 choose a small model for the reachability and size constraints; if there are no models then  $\varphi$  is unsatisfiable
- 2 build a  $\text{FO}(\mathbb{D}, \mathbb{O}, \mathbb{P})$  formula  $\psi$  for the selected small model
- 3 check the satisfiability of  $\psi$

## Remark

*The complexity of the reduction procedure is  $\text{NP}^{\text{MOILP}}$  when the number of universally quantified variables is fixed.*

## Theorem

*If the satisfiability of the underlying first order logic  $FO(\mathbb{D}, \mathbb{O}, \mathbb{P})$  is decidable, then the satisfiability of CSL is decidable*

## Theorem

*For any basic statement  $S$  and any CSL formula  $\varphi$ , we can compute in polynomial time a formula  $\text{post}(S, \varphi)$  describing the strongest post-condition of  $\varphi$  by  $S$ .*

Verified algorithms include

- searching an element into a list of list
- insertion into an array of doubly-linked list

- 1 General logic: gCSL
- 2 CSL
- 3 Decision procedure for CSL
- 4 Related work**
- 5 Conclusions and future work

- finite data domains: the logic LRP from [Yorsh et al. 2007] can specify unbounded trees but it cannot specify lists of doubly-linked lists which is possible in CSL
- abstraction techniques: [Berdine et al. 2007] introduce predicate abstractions based on separation logic formulas. Our approach uses precise invariant checking to reason about composite data structures.
- unbounded data domains:
  - [Bouajjani et al.2007] allows to reason about words and it is included in CSL
  - [Bradley et al.2006] allows to reason about arrays
  - [Habermehl et al.2008] allows more expressive constraints on the index variables but it considers only arrays with integer data
  - [Lahiri et al. 2008] starts from a partial order on the types of the program such that it is not possible to have links going from a smaller type to a greater one. It does not handle the length of the lists nor the arrays.

- 1 General logic: gCSL
- 2 CSL
- 3 Decision procedure for CSL
- 4 Related work
- 5 Conclusions and future work**

# Conclusions and future work

We introduce a logic called **CSL** that:

- is able to express
  - properties on the shape of the heap
  - properties on the size of the heap
  - properties on the data stored in the heap
- it is closed under post-computation
- it has a decidable satisfiability problem

Future work:

- 1 efficient implementation of the decision procedure on top of existing SMT solvers
- 2 use CSL in other verification methods (abstract interpretation, termination proofs)