LOIS: infinite sets in practice

Eryk Kopczyński

Uniwersytet Warszawski

Apr 26, 2014

Example: test whether a graph is connected (BFS)

function $testConnected(V, E)$
  $connected :=$ **true**;
  for $v \in V$
    $R := \{v\}$;
    for $w \in R$ **for** $x \in V$
      if $(w, x) \in E$ **and** $x \notin R$
        R+=$\{x\}$;
    if $R \neq V$
      $connected :=$ **false**;
  return $connected$;

Example: test whether a graph is connected (BFS)

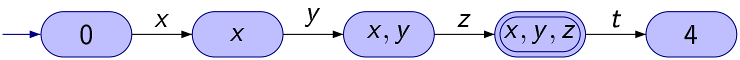
... on a random bipartite graph

$V := \ldots;$
$B :=$ random symmetric relation on V;
$P :=$ random partition of V;
$E := \emptyset$;
for $x \in V$ **for** $y \in V$
  if $(x, y) \in B$ **and** $P(x) \neq P(y)$
    E+=(x, y);
if $testConnected(V, E) \ldots$

Example: minimalization of an automaton

Orbit finite automata (M. Bojańczyk et al.)

alphabet: $\mathbb{A}$



$x, y, z, t \in \mathbb{A}$

$$F = \{(x, y, z) : x = y \vee y = z \vee z = x\}$$

Mikołaj Bojańczyk, Szymon Toruńczyk, FSTTCS 2012
Imperative Programming in Sets with Atoms

What is LOIS?

We want to have a programming language allowing to write programs like in the examples above.

LOIS (Looping Over Infinite Sets) is implemented as a C++11 library, which allows us to easily combine the infinite sets with all the power of C++.

(live show)

LOIS imlementation – an example

As usual, we have an execution stack, for storing function calls and local variables.

In LOIS, we also push the variables ($\in \mathbb{A}$) and the constraints on them.

Instructions are executed in *pseudoparallel* for all the valuations of variables which satisfy the constaints.

```
lset C;
for(elem a: A) {
  lset D;
  for(elem b: A)
    If(a != b)
      D += newSet(b);
  C += D;
  }
cout << "C=" << C << endl;
```

(how the stack contents change, on the board)

LOIS imlementation – an example

As usual, we have an execution stack, for storing function calls and local variables.

In LOIS, we also push the variables ($\in \mathbb{A}$) and the constraints on them.

Instructions are executed in *pseudoparallel* for all the valuations of variables which satisfy the constaints.

```
lset C;
for(elem a: A) {
  lset D;
  for(elem b: A)
    If(a != b)
      D += newSet(b);
  C += D;
  }
cout << "C=" << C << endl;
```

(how the stack contents change, on the board)

More formally: varsets

Varset = a set of variables from $\mathbb{A}$ and first order constraints on them

$$\text{concrete varset } V = \begin{array}{|c|}\hline x \neq y \\ \hline x \in \mathbb{A} \\ \hline y \in \mathbb{A} \\ \hline \end{array}$$

$$V_1 = \begin{array}{|c|}\hline x \neq z \\ \hline z \in \mathbb{A} \\ \hline \end{array} \text{ extends } V$$

$\mathrm{Val}(V)$ – the set of all valuations of V
$\mathrm{Val}_v(V_1)$ – the set of all valuations of V_1 extending $v \in \mathrm{Val}(V)$

more formally: sets and elements

$$B = \{(x, y) | x, y \in \mathbb{A}, x \neq y\}$$

set: $\{e|V\}$

e - V-element (atom, tuple, set, ...)

...

more formally: sets and elements

$$B_y = \{(x, y) | x \in \mathbb{A}, x \neq y\}$$

V-set: $\{e | V_1\}$

e - $V \cup V_1$-element (atom, tuple, set, ...)

...

More formally: sets and elements

$$B = \{1, 2, 3\}$$

simple V-set: $\{e | V_1\}$

e - $V \cup V_1$-element (atom, tuple, set, ...)

V-set = union of a finitely many simple V-sets

More formally: stack and pseudoparallel execution

We push varsets on the stack.

Interpretation:
The program executes in pseudoparallel for each valuation of the varset on the stack.

More formally: instructions

Loop for$(x \in V)$:

- iterates over simple sets,
- for a simple set $\{e|V\}$, it pushes the varset V on the stack, and sets x to e (creating new variable names)

Declaration A:

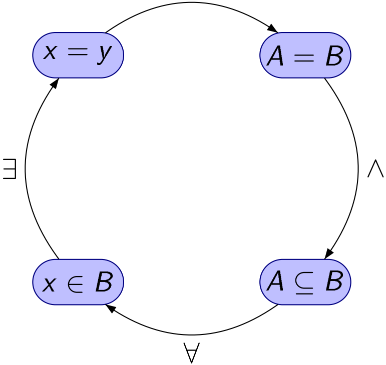
- we create $A_{[v]}$, a copy of A for each $v \in \mathrm{Val}(V)$, where V is the varset on the execution stack

Assignment $A += e$:

- Let $V \cup V_1$ be the varset on the stack at the time of assignment
- For each $v \in \mathrm{Val}(V)$, $A_{[v]}$ is extended by $e_{[v']}$ for each $v' \in \mathrm{Val}_v(V_1)$

We need to decide FO

Conditions are translated to first order formulae over $\mathbb{A}$



We need to decide FO

- Conditions are translated to first order formulae over $\mathbb{A}$
- Is the given formula satisfiable?
- Simplification of formulae, elimination of unnecessary variables

$$
\begin{array}{l}
B := \emptyset; \\
\textbf{for } x \in A \textbf{ for } y \in A \\
\qquad \textbf{if } x = y \\
\qquad\qquad B += (x, y); \\
A := B;
\end{array}
$$

More structure

The set $\mathbb{A}$ can have an additional logical structure:

- order
- random partition
- random binary relation
- homogenuous tree

these are homogenuous structures with finite extension property

- **homogenuous**: each isomorphism of substructures can be extended to an automorphism of $\mathbb{A}$
- **finite extensions**: if $\mathbb{B}$ is a finite substructure of $\mathbb{A}$, it can be extended by 1 element in only finitely many ways (up to isomorphism)

More structure

These two properties ensure that the first order logic is decidable on our structures.

In general, the pseudoparalell semantics of LOIS can be extended to any structure with decidable FO theory. The program will work in finite time, as long as the recursion and iteration depth is bounded.

Singleton construction

Question

what is the power of $\{x, y\}$?

Question

what is the power of $\{x, y\}$?

Answer

$$\mathbb{P}(\{1|x = y\} \cup \{2|x \neq y\})$$

Summary

- example: connectedness of a random bipartite graph
- example: minimalization of an orbit finite automaton
- live show
- execution stack on the board
- pseudoparalell computation more formally
- the necessity of evaluation and simplification of formulae
- additional structures possible
- singleton construction

Thank you!