LOIS: infinite sets in practice

Eryk Kopczyński

Uniwersytet Warszawski

Apr 26, 2014
Example: test whether a graph is connected (BFS)

```plaintext
function testConnected(V, E)
    connected := true;
    for v ∈ V
        R := {v};
        for w ∈ R for x ∈ V
            if (w, x) ∈ E and x ∉ R
                R+={x};

            if R ≠ V
                connected := false;
        return connected;
```
Example: test whether a graph is connected (BFS)

... on a random bipartite graph

\[ V := \ldots ; \]
\[ B := \text{random symmetric relation on } V; \]
\[ P := \text{random partition of } V; \]
\[ E := \emptyset ; \]
\[ \text{for } x \in V \text{ for } y \in V \]
\[ \quad \text{if } (x, y) \in B \text{ and } P(x) \neq P(y) \]
\[ \quad E += (x, y); \]
\[ \text{if } \text{testConnected}(V, E) \ldots \]
Example: minimalization of an automaton

Orbit finite automata (M. Bojańczyk et al.)

alphabet: $\mathbb{A}$

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

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)*
As usual, we have an execution stack, for storing function calls and local variables.
In LOIS, we also push the variables \((\in A)\) and the constraints on them.
Instructions are executed in pseudoparallel for all the valuations of variables which satisfy the constraints.

```c++
set C;
for (elem a: A) {
    set 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 implementation – an example

As usual, we have an execution stack, for storing function calls and local variables.
In LOIS, we also push the variables \( (\in A) \) and the constraints on them.
Instructions are executed in \textit{pseudoparallel} for all the valuations of variables which satisfy the constraints.

\begin{verbatim}
  lset C;
  for (elem a: A) {
    lset D;
    for (elem b: A)
      if (a != b)
        D += newSet(b);
    C += D;
  }
  cout << "C=" << C << endl;
\end{verbatim}

\textit{(how the stack contents change, on the board)}
More formally: varssets

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

Concrete varset $V = \begin{bmatrix} x \neq y \\ x \in \mathbb{A} \\ y \in \mathbb{A} \end{bmatrix}$

\[ V_1 = \begin{bmatrix} x \neq z \\ z \in \mathbb{A} \end{bmatrix} \] extends $V$

$\text{Val}(V) - \text{the set of all valuations of } V$

$\text{Val}_v(V_1) - \text{the set of all valuations of } V_1 \text{ extending } v \in \text{Val}(V)$
more formally: sets and elements

\[ B = \{(x, y) | x, y \in A, x \neq y\} \]

set: \( \{e | V\} \)

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

...
more formally: sets and elements

\[ B_y = \{(x, y) \mid x \in A, x \neq y\} \]

\[ \text{V-set: } \{e \mid V_1\} \]

\[ e - \text{ } V \cup V_1\text{-element (atom, tuple, set, ...)} \]

\[ \cdots \]
More formally: sets and elements

\[ B = \{1, 2, 3\} \]

simple \( V \)-set: \( \{e \mid V_1\} \)

\( e - V \cup V_1 \)-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 \(v \in \text{Val}(V)\), 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 \(v \in \text{Val}(V)\), \(A_{[v]}\) is extended by \(e_{[v']}\) for each \(v' \in \text{Val}_v(V_1)\)
Conditions are translated to first order formulae over $\mathbb{A}$

\[ A \subseteq B \land \forall x \in B \exists y = x \quad \text{such that} \quad A = B \]
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

$$B := \emptyset;$$
$$\text{for } x \in A \text{ for } y \in A$$
  $$\text{if } x = y$$
  $$B^+ = (x, y);$$

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

- order
- random partition
- random binary relation
- homogenous tree

These are homogenous structures with finite extension property

- **homogenous**: 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)
These two properties ensure that the first order logic is decidable on our structures.

In general, the pseudoparallel 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.
Question
what is the power of \( \{x, y\} \)?
Question
what is the power of \( \{x, y\} \)?

Answer

\[
\mathbb{P} ( \{1| x = y \} \cup \{2| x \neq y \} )
\]
example: connectedness of a random bipartite graph
example: minimalization of an orbit finite automaton
live show
execution stack on the board
pseudoparallel computation more formally
the necessity of evaluation and simplification of formulae
additional structures possible
singleton construction

Thank you!