Sparse Bidirectional Data Flow Analysis as a Basis for Type Inference

Jeremy Singer @ cl cam ac uk

http://www.cl.cam.ac.uk/~jds31/research/typeinf.pdf

Slide 0 - Title slide. Gives URL of draft paper on this topic - http://www.cl.cam.ac.uk/~jds31/research/typeinf.pdf

Data Flow Analysis

Classical style (Dragon Blanket)

Control flow graph

Data flow equations

Generate and solve

1/7

Slide 1 - reviewing classical compiler data flow analysis paradigm

Bidirectional Data Flow Analysis

In which direction (wrt control flow) does data flow info need to propagate?

- forward (e.g. constant prop°)
- backward (e.g. liveness)
- bidirectional (e.g. PRE)

Slide 2 - reviewing bidirectional data flow analysis - information flows with and against control flow

Bidirectional Data Flow Analysis for Type Inference

```
i := a[j]
k := a[i]
```

Proposed by Uday Khedker et al. [KDM03]

Why ?

- "because it's there"
- remove dynamic type checks
- reverse engineer types

Slide 3 - Justification for treating type inference as a bidirectional data flow problem

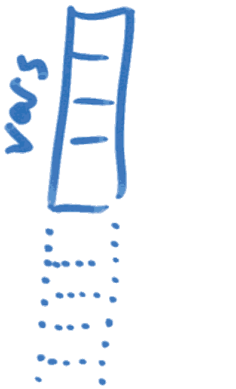
Sparse Data Flow Analysis

- only store info where needed
- only propagate info when needed

(more efficient, more precise than classical dfa)

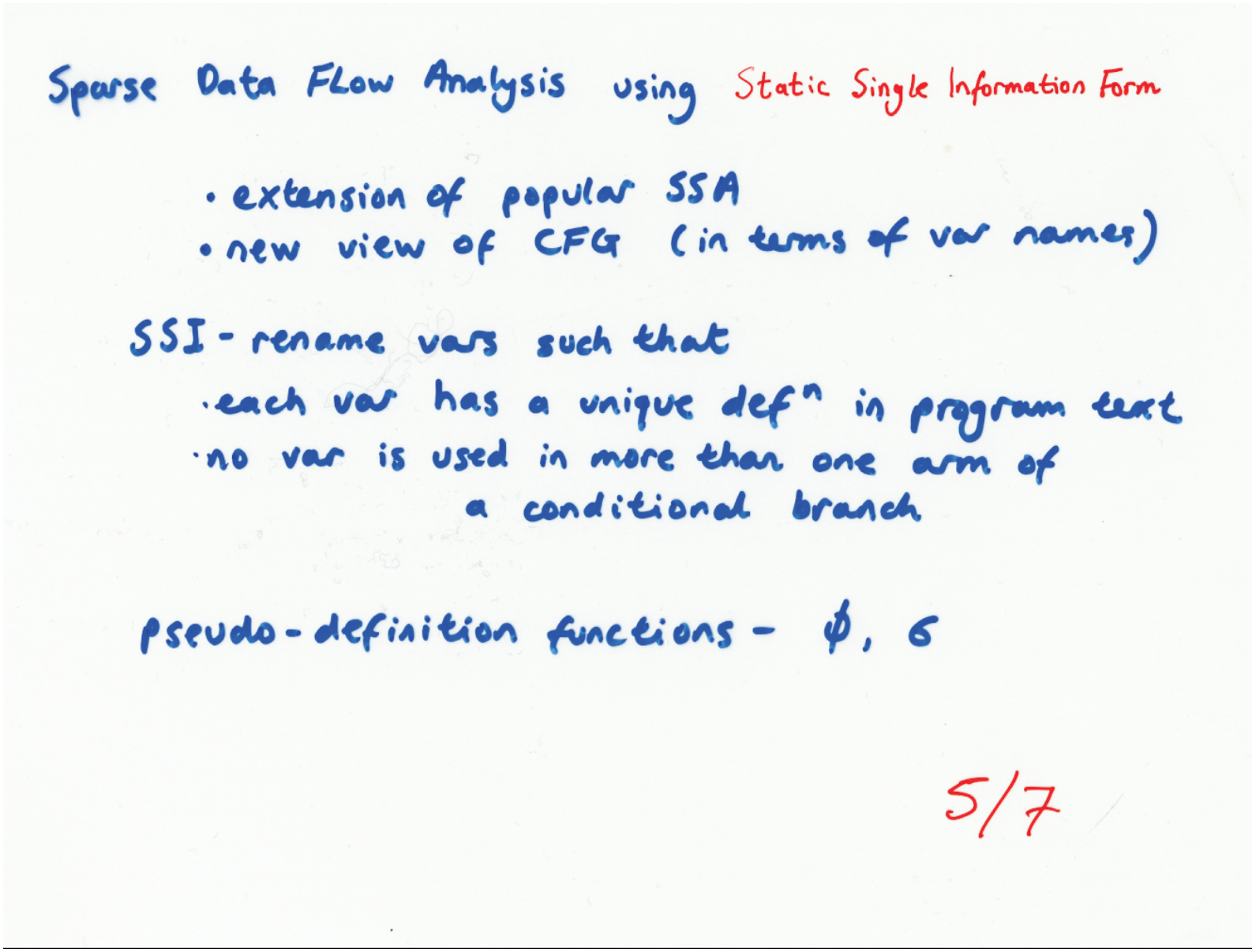
classical

sparse



4/7

Slide 4 - reviewing sparse data flow analysis paradigm, only store data flow info per-variable, rather than per-variable, per-node.

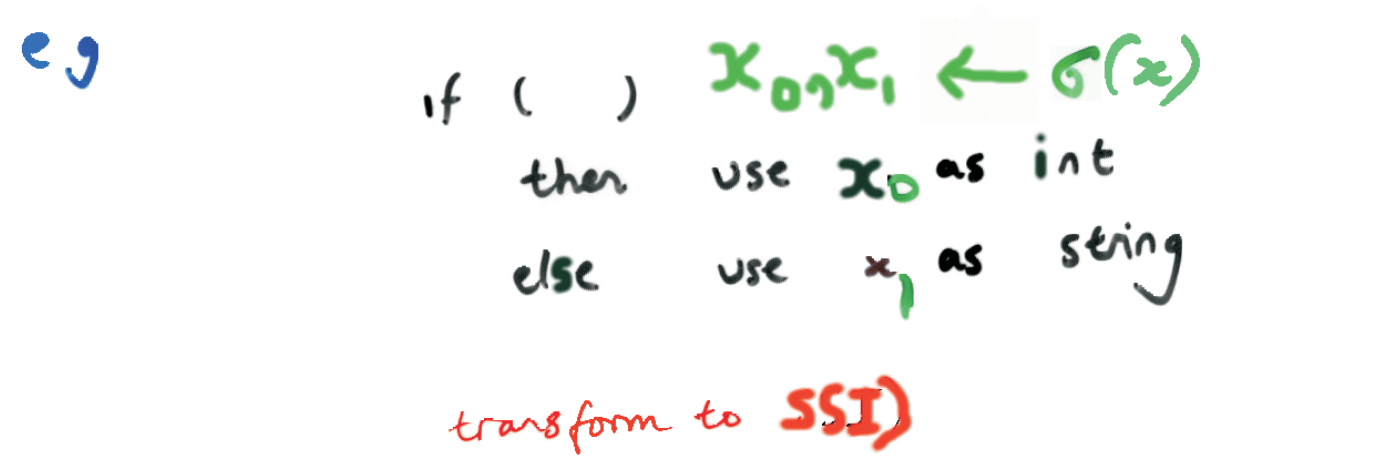


Slide 5 - reviewing static single information form - a sparse representation similar to SSA form

First significant claim

SSI for Type Inference

SSI reduces dynamically typed progs to statically typed progs



Slide 6 - first significant claim. Converting to SSI reduces some dynamically typed vars to statically typed vars.

Second significant claim

Sparse bidirectional data flow analysis for type inference can be performed on SSI progs

- more efficient than classical dfa [KDM03]
- ϕ/σ fns in just the right place for data flow

http://www.cl.cam.ac.uk/~jds31/research/typeinf.pdf

Slide 7 - second significant claim. Converting to SSI allows bidirectional type inference to be performed sparsely. See paper for full details.