Static Program Analysis
The concept of static program analysis

- A static program analysis is a tool (a program) that gives some kind of useful information about the run-time behaviour of any input program, within finite time, without executing the input program.

- Typically, a program analysis can be imprecise.
  - The real property of interest can be undecidable. A program analysis may anyway provide useful information in many cases within a finite period of time.
Examples of classic analysis

• Reaching Definitions –
  detect for all program points which assignments to variables can be in force there.

• Constant Propagation –
  detect for all program points which variables definitely have a constant value there, and find out the constant values.

• Detection of Signs –
  for each program point and each variable, find out all possible signs that the value of the variable can have (+, −, 0).
Examples of useful information provided by static analyses

• Variable at some program point is used before initialization.
  – Possibly provided by Reaching Definitions, if no definition of a variable turns out to be reaching a point of its use.

• Some part of code is never executed.
  – Possibly provided by Constant Propagation, if the condition of a loop turns out to be a constant.

• Index used at an array lookup is out of bounds.
  – Possibly provided by Detection of Signs, if an index turns out to be always negative.
Abstract values and conservative approximation

- An analysis can be specified as a non-standard semantics that associates some abstract values to syntactic objects.

- An analysis must be a conservative approximation of the property of interest. It means that an analysis can err in exactly one direction.
  - This can be specified in terms of an order relation on the set of abstract values of that analysis: the abstract value representing the property of interest can either only be less than or equal to, or only be greater than or equal to, the abstract value provided by the analysis.
  - The allowed direction of erring depends on our needs: which direction is considered a safe approximation.
Kinds of conservative approximation

- **May-analyses** –
  
  the answer may entail incorrect values but never leave correct ones out.

  All three example analyses above are may-analyses (incl. Constant Propagation that basically finds an upper approximation of the set of all possible values of any variable at any program point).

- **Must-analyses** –
  
  the answer must entail only correct values but it occasionally leaves some of them out.
Detection of Signs
The semantic domains

- The complete lattice of abstract arithmetic values (blackboard).
- The complete lattice of abstract boolean values (blackboard).
- The complete lattice of property states (blackboard).
DS semantics of numerals

• The type of the semantic function (blackboard).

• A compositional definition (blackboard).
DS semantics of arithmetic and boolean expressions

- The types of the semantic functions (blackboard).
- Compositional definition forms in the lines of Tables 7.1 and 7.3 (blackboard).
- Definitions of the semantic operations in the lines of Tables 7.2 and 7.4 (oneself).
DS semantics of statements

• The type of the semantic function (blackboard):
  – Transformations of property states are total, as the analysis must not loop.

• The first attempt of a compositional definition in the lines of Table 7.5 (blackboard).

• Example 7.6 (blackboard).
Modified DS semantics of statements

• The second attempt of a compositional definition (blackboard).

• Example 7.7 (blackboard).

• Example 7.8 (blackboard).

• One more alternative definition (blackboard):
  – The principles behind the second definition revealed;
  – Examples of enabling more precise analysis than other definitions;
  – Why is this definition nevertheless rejected?
Correctness of the definition

• There exist semantic functions specified before (blackboard):
  – Lemma 7.11 for the second version of the semantics.
  – Lemma 7.13.
  – Proposition 7.15.
The role of continuity

• Exercise 7.14 (blackboard).

• Comparison to Essential Exercise 5.46 (blackboard).

• Interpretation of $\text{State} \rightarrow \text{State}$ as $\text{State} \cup \{\bot\} \rightarrow \text{State} \cup \{\bot\}$ (blackboard):
  
  – Which functions are continuous?
  
  – Which functions can be expressed in the case of strict or lazy evaluation?

• Continuity of the semantic functions used in our definitions (blackboard):
  
  – Continuity vs monotonicity;
  
  – Proofs.
Safety of the analysis

• Essential Exercise 7.18 (blackboard).

• Essential Exercise 7.19 (blackboard).

• Theorem 7.20 (blackboard):
  – Lemma 7.21;
  – Lemma 7.22
Program transformation

- Assertion forms (blackboard).
- Definition of validity (blackboard).
- Some valid transformations (blackboard):
  - Trivial example of identity;
  - Lemma 7.25;
  - Exercise 7.26;
  - Exercise 7.27;
  - Exercise 7.28;
  - Exercise 7.29.
Exercise

• For the program

```
if x*y = 0
  then (x := x + 1 ; y := y - 1)
else if x = 0 then x := x + 1 else x := x - 1
```

find a non-trivial valid transformation detectable the Detection of Signs analysis and build a derivation tree for it (blackboard).
Exercise

- For the program

\[
(x := 0 ; a := 1) ; \\
\textbf{if} \ a \leq x \\
\quad \textbf{then} \ x := 2 \times x \\
\quad \textbf{else} \ (x := x + a ; a := a - 1)
\]

find a non-trivial valid transformation detectable by the Detection of Signs analysis and build a derivation tree for it (oneself).
Constant Propagation
The semantic domains

- The complete lattice of abstract arithmetic values (blackboard).
- The complete lattice of abstract boolean values (blackboard).
- The complete lattice of property states (blackboard).
CP semantics of numerals

- The type of the semantic function (blackboard).
- A compositional definition (oneself).
CP semantics of arithmetic and boolean expressions

- The types of the semantic functions (blackboard).
- Compositional definition forms (oneself).
- Definitions of the semantic operations (oneself).
CP semantics of statements

• The type of the semantic function (blackboard).

• A compositional definition (oneself).

• Exercise 7.17 (home).

• Exercise 7.24 (home).
7 Static Program Analysis
7.4 Data Flow Frameworks

Data Flow Frameworks
Kinds of analyses according to the approach

• Forward analyses –
  computation proceeds in the same direction as in standard denotational semantics, deducing properties of outputs from the properties of inputs.

• Backward analyses –
  computation proceeds in the direction opposite to that in standard denotational semantics, deducing properties of inputs from the properties of outputs.
Data flow frameworks

• Properties and property states (blackboard).

• Forward analyses (blackboard):
  – Types of semantic functions corresponding to forward analyses.
  – Definition schemata.

• Backward analyses (blackboard):
  – Types of semantic functions corresponding to forward analyses.
  – Definition schemata.