

Static Program Analysis

Lecture 13: Abstract Interpretation II (Abstract Semantics of WHILE)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

`noll@cs.rwth-aachen.de`

`http://www-i2.informatik.rwth-aachen.de/i2/spa11/`

Summer Semester 2011

EINLADUNG

Zeit: Mittwoch, 01. Juni 2011, 15.00 Uhr
Ort: AH IV, Ahornstr. 55
Referentin: Frau Prof. Dr. Barbara M. Terhal,
Institute for Quantum Information, FB1,
RWTH Aachen
Titel: Quantum Complexity Theory

Abstract:

We will discuss the model of quantum computation and the important differences between quantum and classical computation. We then review the quantum version of the classical Cook-Levin theorem due to Kitaev which proves that a quantum (or matrix) version of the satisfiability problem is quantum NP or QMA-complete. We give an overview of more recent advances in quantum complexity theory and discuss various open questions.

Es laden ein: Die Dozenten der Informatik

- 1 Repetition: Abstract Interpretation
- 2 Repetition: Concrete Semantics of WHILE Programs
- 3 More on Concrete Semantics
- 4 Abstract Semantics

- **Summary:** a theory of **sound approximation** of the semantics of programs
- **Basic idea:** execution of program on **abstract values** (similar to type-level bytecode interpreter)
- **Example:** parity (even/odd) rather than concrete numbers
- **Procedure:** run program on finite set of abstract values that **cover all concrete inputs** using abstract operations that **cover all concrete outputs**
 $\implies$ **soundness** of approach
- **Preciseness** of information again characterized by **partial order**

Definition (Galois connection)

Let $(L, \sqsubseteq_L)$ and $(M, \sqsubseteq_M)$ be complete lattices. A pair (α, γ) of monotonic functions

$$\alpha : L \rightarrow M \quad \text{and} \quad \gamma : M \rightarrow L$$

is called a **Galois connection** if

$$\forall l \in L : l \sqsubseteq_L \gamma(\alpha(l)) \quad \text{and} \quad \forall m \in M : \alpha(\gamma(m)) \sqsubseteq_M m$$

Interpretation:

- $L = \{\text{sets of concrete values}\}$, $M = \{\text{sets of abstract values}\}$
- $\alpha = \text{abstraction function}$, $\gamma = \text{concretization function}$
- $l \sqsubseteq_L \gamma(\alpha(l))$: α yields over-approximation
- $\alpha(\gamma(m)) \sqsubseteq_M m$: no loss of precision by abstraction after concretization
- Usually: $l \neq \gamma(\alpha(l))$, $\alpha(\gamma(m)) = m$

- 1 Repetition: Abstract Interpretation
- 2 Repetition: Concrete Semantics of WHILE Programs
- 3 More on Concrete Semantics
- 4 Abstract Semantics

Definition (Execution relation for statements)

If $c \in \text{Cmd}$ and $\sigma \in \Sigma$, then $\langle c, \sigma \rangle$ is called a **configuration**. The **execution relation** (on configurations and states) is defined by the following rules:

$$\text{(skip)} \frac{}{\langle \text{skip}, \sigma \rangle \rightarrow \sigma}$$

$$\text{(asgn)} \frac{}{\langle x := a, \sigma \rangle \rightarrow \sigma[x \mapsto \text{val}_\sigma(a)]}$$

$$\text{(seq1)} \frac{\langle c_1, \sigma \rangle \rightarrow \langle c'_1, \sigma' \rangle}{\langle c_1 ; c_2, \sigma \rangle \rightarrow \langle c'_1 ; c_2, \sigma' \rangle}$$

$$\text{(seq2)} \frac{\langle c_1, \sigma \rangle \rightarrow \sigma'}{\langle c_1 ; c_2, \sigma \rangle \rightarrow \langle c_2, \sigma' \rangle}$$

Definition (Execution relation for statements; continued)

$$\text{(if1)} \frac{val_{\sigma}(b) = \text{true}}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \langle c_1, \sigma \rangle}$$

$$\text{(if2)} \frac{val_{\sigma}(b) = \text{false}}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \langle c_2, \sigma \rangle}$$

$$\text{(wh1)} \frac{val_{\sigma}(b) = \text{true}}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \langle c; \text{while } b \text{ do } c, \sigma \rangle}$$

$$\text{(wh2)} \frac{val_{\sigma}(b) = \text{false}}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma}$$

- 1 Repetition: Abstract Interpretation
- 2 Repetition: Concrete Semantics of WHILE Programs
- 3 More on Concrete Semantics**
- 4 Abstract Semantics

Example 13.1

- $c := y := 1; \underbrace{\text{while } \neg(x=1)}_b \text{ do } \underbrace{y := y*x}_{c_1}; \underbrace{x := x-1}_{c_2}$
 $\underbrace{\hspace{10em}}_{c_0}$
- Claim: $\langle c, \sigma \rangle \rightarrow^+ \sigma_{1,6}$ for every $\sigma \in \Sigma$ with $\sigma(\mathbf{x}) = 3$
- Notation: $\sigma_{i,j}$ means $\sigma(\mathbf{x}) = i, \sigma(\mathbf{y}) = j$
- Derivation: on the board

This operational semantics is well defined in the following sense:

Theorem 13.2

*The execution relation for statements is **deterministic**, i.e., whenever $c \in \text{Cmd}$, $\sigma \in \Sigma$ and $\kappa_1, \kappa_2 \in \text{Cmd} \times \Sigma \cup \Sigma$ such that $\langle c, \sigma \rangle \rightarrow \kappa_1$ and $\langle c, \sigma \rangle \rightarrow \kappa_2$, then $\kappa_1 = \kappa_2$.*

This operational semantics is well defined in the following sense:

Theorem 13.2

*The execution relation for statements is **deterministic**, i.e., whenever $c \in \text{Cmd}$, $\sigma \in \Sigma$ and $\kappa_1, \kappa_2 \in \text{Cmd} \times \Sigma \cup \Sigma$ such that $\langle c, \sigma \rangle \rightarrow \kappa_1$ and $\langle c, \sigma \rangle \rightarrow \kappa_2$, then $\kappa_1 = \kappa_2$.*

Proof.

omitted □

- 1 Repetition: Abstract Interpretation
- 2 Repetition: Concrete Semantics of WHILE Programs
- 3 More on Concrete Semantics
- 4 Abstract Semantics

Definition 13.3

Let (α, γ) be a Galois connection with $\alpha : L \rightarrow M$ and $\gamma : M \rightarrow L$, and let $f : L^n \rightarrow L$ and $f^\# : M^n \rightarrow M$ be functions of rank $n \in \mathbb{N}$. Then $f^\#$ is called a **safe approximation** of f if, whenever $m_1, \dots, m_n \in M$,

$$\alpha(f(\gamma(m_1), \dots, \gamma(m_n))) \sqsubseteq_M f^\#(m_1, \dots, m_n).$$

Moreover it is called **most precise** safe approximation if the reverse inclusion is also true.

- **Interpretation:** the abstraction $f^\#$ of f covers all concrete results
- **Note:** monotonicity of f and/or $f^\#$ is *not* required (but usually given; see Lemma 13.5)

Example 13.4

① Parity abstraction (cf. Example 12.2): most precise approximations

- $n = 0$: $1^\# = \{\text{odd}\}$
- $n = 1$: $-^\#(P) = P$, $(-1)^\#(\{\text{even}\}) = \{\text{odd}\}$
- $n = 2$: $\{\text{even}\} +^\# \{\text{odd}\} = \{\text{odd}\}$, $\{\text{even}\} \cdot^\# \{\text{odd}\} = \{\text{even}\}$

Example 13.4

- ① Parity abstraction (cf. Example 12.2): most precise approximations
 - $n = 0$: $1^\# = \{\text{odd}\}$
 - $n = 1$: $-^\#(P) = P$, $(-1)^\#(\{\text{even}\}) = \{\text{odd}\}$
 - $n = 2$: $\{\text{even}\} +^\# \{\text{odd}\} = \{\text{odd}\}$, $\{\text{even}\} \cdot^\# \{\text{odd}\} = \{\text{even}\}$
- ② Sign abstraction (cf. Example 12.3): most precise approximations
 - $n = 0$: $1^\# = \{+\}$
 - $n = 1$: $-^\#(\{+\}) = \{-\}$, $(-1)^\#(\{+\}) = \{+, 0\}$
 - $n = 2$: $\{+\} +^\# \{+\} = \{+\}$, $\{+\} +^\# \{-\} = \{+, -, 0\}$
 $\{+\} \cdot^\# \{-\} = \{-\}$

Example 13.4

- ① Parity abstraction (cf. Example 12.2): most precise approximations
 - $n = 0$: $1^\# = \{\text{odd}\}$
 - $n = 1$: $-^\#(P) = P$, $(-1)^\#(\{\text{even}\}) = \{\text{odd}\}$
 - $n = 2$: $\{\text{even}\} +^\# \{\text{odd}\} = \{\text{odd}\}$, $\{\text{even}\} \cdot^\# \{\text{odd}\} = \{\text{even}\}$
- ② Sign abstraction (cf. Example 12.3): most precise approximations
 - $n = 0$: $1^\# = \{+\}$
 - $n = 1$: $-^\#(\{+\}) = \{-\}$, $(-1)^\#(\{+\}) = \{+, 0\}$
 - $n = 2$: $\{+\} +^\# \{+\} = \{+\}$, $\{+\} +^\# \{-\} = \{+, -, 0\}$
 $\{+\} \cdot^\# \{-\} = \{-\}$
- ③ Interval abstraction (cf. Example 12.4): most precise approximations
 - $n = 0$: $z^\# = [z, z]$
 - $n = 1$: $-^\#([z_1, z_2]) = [-z_2, -z_1]$, $(-1)^\#([z_1, z_2]) = [z_1 - 1, z_2 - 1]$
 - $n = 2$:
 - $[y_1, y_2] +^\# [z_1, z_2] = [y_1 + z_1, y_2 + z_2]$
 - $[y_1, y_2] -^\# [z_1, z_2] = [y_1 - z_2, y_2 - z_1]$

Lemma 13.5

If $f : L^n \rightarrow L$ and $f^\# : M^n \rightarrow M$ are monotonic, then $f^\#$ is a safe approximation of f iff, for all $l_1, \dots, l_n \in L$,

$$\alpha(f(l_1, \dots, l_n)) \sqsubseteq_M f^\#(\alpha(l_1), \dots, \alpha(l_n)).$$

Lemma 13.5

If $f : L^n \rightarrow L$ and $f^\# : M^n \rightarrow M$ are monotonic, then $f^\#$ is a safe approximation of f iff, for all $l_1, \dots, l_n \in L$,

$$\alpha(f(l_1, \dots, l_n)) \sqsubseteq_M f^\#(\alpha(l_1), \dots, \alpha(l_n)).$$

Proof.

on the board



- **Reminder:** concrete semantics of WHILE
 - **states** $\Sigma := \{\sigma \mid \sigma : Var \rightarrow \mathbb{Z}\}$ (Definition 12.6)
 - **execution relation** $\rightarrow \subseteq (Cmd \times \Sigma) \times (Cmd \times \Sigma \cup \Sigma)$ (Definition 12.9)

- **Reminder:** concrete semantics of WHILE
 - **states** $\Sigma := \{\sigma \mid \sigma : \text{Var} \rightarrow \mathbb{Z}\}$ (Definition 12.6)
 - **execution relation** $\rightarrow \subseteq (\text{Cmd} \times \Sigma) \times (\text{Cmd} \times \Sigma \cup \Sigma)$ (Definition 12.9)
- Yields **concrete domain** $L := 2^\Sigma$ and concrete transition function:

Definition 13.6 (Concrete transition function)

The **concrete transition function** of WHILE is defined by the family of functions

$$\text{next}_{c,c'} : 2^\Sigma \rightarrow 2^\Sigma$$

where $c \in \text{Cmd}$, $c' \in \text{Cmd} \cup \{\downarrow\}$ and, for every $S \subseteq \Sigma$,

$$\begin{aligned}\text{next}_{c,c'}(S) &:= \{\sigma' \in \Sigma \mid c' \in \text{Cmd}, \exists \sigma \in S : \langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle\} \text{ and} \\ \text{next}_{c,\downarrow}(S) &:= \{\sigma' \in \Sigma \mid \exists \sigma \in S : \langle c, \sigma \rangle \rightarrow \sigma'\}\end{aligned}$$

Safe Approximation of Execution Relation II

- **Reminder:** abstraction determined by **Galois connection** (α, γ) with $\alpha : L \rightarrow M$ and $\gamma : M \rightarrow L$
 - here: $L := 2^\Sigma$, M not fixed (usually $M = Var \rightarrow \dots$ or $M = 2^{Var \rightarrow \dots}$)
 - write Abs in place of M
 - thus $\alpha : 2^\Sigma \rightarrow Abs$ and $\gamma : Abs \rightarrow 2^\Sigma$

Safe Approximation of Execution Relation II

- **Reminder:** abstraction determined by **Galois connection** (α, γ) with $\alpha : L \rightarrow M$ and $\gamma : M \rightarrow L$
 - here: $L := 2^\Sigma$, M not fixed (usually $M = \text{Var} \rightarrow \dots$ or $M = 2^{\text{Var} \rightarrow \dots}$)
 - write Abs in place of M
 - thus $\alpha : 2^\Sigma \rightarrow Abs$ and $\gamma : Abs \rightarrow 2^\Sigma$
- Yields abstract semantics:

Definition 13.7 (Abstract semantics of WHILE)

Given $\alpha : 2^\Sigma \rightarrow Abs$, an **abstract semantics** is defined by a family of functions

$$\text{next}_{c,c'}^\# : Abs \rightarrow Abs$$

where $c \in \text{Cmd}$, $c' \in \text{Cmd} \cup \{\downarrow\}$, and each $\text{next}_{c,c'}^\#$ is a safe approximation of $\text{next}_{c,c'}$, i.e.,

$$\alpha(\text{next}_{c,c'}(\gamma(abs))) \sqsubseteq_{Abs} \text{next}_{c,c'}^\#(abs)$$

for every $abs \in Abs$. Notation:

- $\langle c, abs \rangle \Rightarrow \langle c', abs' \rangle$ for $\text{next}_{c,c'}^\#(abs) = abs'$ and
- $\langle c, abs \rangle \Rightarrow abs'$ for $\text{next}_{c,\downarrow}^\#(a) = abs'$