Software Verification - Few Notations

Jérôme Leroux

Univ. Bordeaux & CNRS, LaBRI, UMR 5800, Talence, France
leroux@labri.fr

Abstract. In this lesson we provide a quick overview of the main notations used in the sequel.

1 Expressions and Logics

Let V be a countable set of elements, called *variables*. A *valuation* of V is a function $\rho : V \mapsto \mathbb{Z}$ where $\mathbb{Z}$ is the set of integers. Given $x \in V$ and $n \in \mathbb{Z}$, we denote by $\rho[x := n]$ the valuation ρ' defined by $\rho'(v) = \rho(v)$ for every $v \in V \backslash \{x\}$ and $\rho'(x) = n$. An *expression* on V is a *term* e obtained with the following grammar where $v \in V$ and $n \in \mathbb{Z}$:

$$e, e_1, e_2 \quad := \quad v \mid n \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 * e_2 \mid e_1 \,/\, e_2$$

Valuations $\rho : V \mapsto \mathbb{Z}$ are extended into the unique (partial) function over the expressions on V satisfying for every $n \in \mathbb{Z}$ and e_1, e_2 expressions on V:

- $\rho(n) = n$,
- $\rho(e_1 + e_2) = \rho(e_1) + \rho(e_2)$,
- $\rho(e_1 - e_2) = \rho(e_1) - \rho(e_2)$,
- $\rho(e_1 * e_2) = \rho(e_1) * \rho(e_2)$, and
- $\rho(e_1 \,/\, e_2) = \mathrm{cdiv}(\rho(e_1), \rho(e_2))$ if $\rho(e_2) \neq 0$.

The integer $\mathrm{cdiv}(n, m)$ where n, m are two integers with $m \neq 0$ is the result of the division of n by m rounded towards zero. This is indeed the behavior of division in most programming languages (including C). Denoting by $\lceil x \rceil$ and $\lfloor x \rfloor$ respectively the ceiling and the floor of a real number x, the value $\mathrm{cdiv}(n, m)$ when $m \neq 0$ is formally defined as follows:

$$\mathrm{cdiv}(n, m) = \begin{cases} \lfloor \frac{n}{m} \rfloor & \text{if } \frac{n}{m} \geq 0 \\ \lceil \frac{n}{m} \rceil & \text{if } \frac{n}{m} < 0 \end{cases}$$

The set of *variables* $\mathrm{var}(e)$ of a term e is defined inductively as follows for every $n \in \mathbb{Z}$, $x \in V$ and e_1, e_2 expressions on V:

- $\mathrm{var}(n) = \emptyset$,
- $\mathrm{var}(x) = \{x\}$,
- $\mathrm{var}(e_1 \sim e_2) = \mathrm{var}(e_1) \cup \mathrm{var}(e_2)$ for every $\sim \in \{+, -, *, /\}$.

Example 1.1. Assume that $e = v + 10$ then $\mathrm{var}(e) = \{v\}$ and $\rho(e) = \rho(v) + 10$ for every valuation ρ defined on v.

A formula ψ on V is a term obtained with the following grammar where e_1, e_2 are expressions on V, and $\sim \in \{\leq, <, =, >, \geq\}$:

$$\psi, \psi_1, \psi_2 \quad := \quad \mathbf{true} \mid \mathbf{false} \mid e_1 \sim e_2 \mid \neg\psi_1 \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2$$

A formula of the form $e_1 \sim e_2$ is called a *simple formula* or a *guard*.

The relation $\rho \models \psi$ where ρ is a valuation on V, and ψ is a formula on V is defined inductively on formulas by $\rho \models \psi$ if one of the following conditions hold:

- $\psi = \mathbf{true}$, or
- $\psi = (e_1 \sim e_2)$ and $\rho(e_1) \sim \rho(e_2)$, or
- $\psi = (\neg\psi_1)$ and not $\rho \models \psi_1$, or
- $\psi = (\psi_1 \vee \psi_2)$ and $\rho \models \psi_1 \vee \rho \models \psi_2$, or
- $\psi = (\psi_1 \wedge \psi_2)$ and $\rho \models \psi_1 \wedge \rho \models \psi_2$.

We also denote by $\rho \not\models \psi$ if not $\rho \models \psi$. When $\rho \models \psi$ we say that ρ satisfies ψ, or ρ is a model of ψ. The set of models of ψ is denoted by $[\![\psi]\!]$.

We associate to a finite set of variables X, multiple copies indexed by $i \in \mathbb{Z}$. Formally, the i-th copy of $x \in X$ is the pair (i, x), usually denoted by $x^{(i)}$. We also introduce the set $X^{(i)} = \{x^{(i)} \mid x \in X\}$. Let $V = \bigcup_{i\in\mathbb{Z}} X^{(i)}$ be the set of copies and let $j \in \mathbb{Z}$. We associate to a variable $v = x^{(i)}$ in V the variable $v^{(j)}$ defined as $x^{(i+j)}$. This transformation is extended on expressions e on V by replacing inductively variables $v \in V$ by $v^{(j)}$. The resulting expression is denoted by $e^{(j)}$. The transformation is also extended on formulas on V in the same way and it produces the formula $\psi^{(j)}$ from a formula ψ. The set $X^{(0)}$ and X are identified, and X' denotes $X^{(1)}$.

Example 1.2. Assume that $\psi = (x' = x + 2)$ then $\psi^{(1)} = (x^{(2)} = x' + 2)$.

2 Operations and control flow automata (CFA)

Let X be an implicit finite set of variables. An *operation* $\mathtt{op}$ is either a *guard* g where g is a simple formula on X, or an *assignement* $x := e$ where $x \in X$ and e is an expression on X. The semantics of an operation $\mathtt{op}$ is a formula $\ll \mathtt{op} \gg$ on $X \cup X'$ defined as follows:

$$\begin{aligned} \ll g \gg \quad &= \quad g \wedge \bigwedge_{z\in X} z' = z \\ \ll x := e \gg \quad &= \quad (x' = e) \wedge \mathrm{side}(e) \wedge \bigwedge_{z\in X, z\neq x} z' = z \end{aligned}$$

where $\mathrm{side}(e)$ is a formula defined inductivelly on the structure of the expressions by:

– $\mathrm{side}(n) = \mathbf{true}$ for every $n \in \mathbb{Z}$,
– $\mathrm{side}(e_1 + e_2) = \mathrm{side}(e_1) \wedge \mathrm{side}(e_2)$,
– $\mathrm{side}(e_1 - e_2) = \mathrm{side}(e_1) \wedge \mathrm{side}(e_2)$,
– $\mathrm{side}(e_1 * e_2) = \mathrm{side}(e_1) \wedge \mathrm{side}(e_2)$, and
– $\mathrm{side}(e_1 \,/\, e_2) = \mathrm{side}(e_1) \wedge \mathrm{side}(e_2) \wedge \neg e_2 = 0$.

The set of operations is denoted by $\mathtt{Op}$. Given an operation $\mathtt{op}$, we denote by $\mathrm{Post}_{\mathtt{op}} : (\mathcal{P}(\mathbb{Z}^X), \subseteq) \mapsto (\mathcal{P}(\mathbb{Z}^X), \subseteq)$ the function defined for every $S \subseteq \mathbb{Z}^X$ by:

$$\mathrm{Post}_{\mathtt{op}}(S) = \begin{cases} S \cap [\![g]\!] & \text{if } \mathtt{op} = (g) \\ \{\rho[x := \rho(e)] \mid \rho \in S\} & \text{if } \mathtt{op} = (x := e) \end{cases}$$

Lemma 2.1. *We have $\rho' \in \mathrm{Post}_{\mathtt{op}}(\{\rho\})$ if, and only if, $r \models\!\!\ll \mathtt{op} \gg$ where r is the valuation on $X \cup X'$ defined by $r(x) = \rho(x)$ and $r(x') = \rho'(x)$ for every $x \in X$.*

A *control flow automaton (CFA for short)* is a tuple $(Q, q_{\mathrm{ini}}, q_{\mathrm{bad}}, \Delta)$ where Q is a non-empty finite set of *control states*, q_{ini} is the *initial control state*, q_{bad} is the *final one*, and $\Delta \subseteq Q \times \mathtt{Op} \times Q$ is a finite ste of *transitions*. A *trace* from a control state q to a control state q' labeled by a word w of operations is a word of transitions $\delta_1 \dots \delta_k$ such that there exists $q_0, \dots, q_k \in Q$ and operations $\mathtt{op}_1, \dots, \mathtt{op}_k \in \mathtt{Op}$ such that $q_0 = q$, $q_k = q'$ and $\delta_j = (q_{j-1}, \mathtt{op}_j, q_j)$. A *configuration* is a pair (q, ρ) where $q \in Q$ and ρ is a valuation of X. When $q = q_{\mathrm{ini}}$ the configuration is said to be *initial*, and when $q = q_{\mathrm{bad}}$ the configuration is said to be *bad*. We associate to an operation $\mathtt{op}$ the binary relation $\xrightarrow{\mathtt{op}}$ over the configurations defined by $(q, \rho) \xrightarrow{\mathtt{op}} (q', \rho')$ if $(q, \mathtt{op}, q') \in \Delta$ and $\rho' \in \mathrm{Post}_{\mathtt{op}}(\{\rho\})$. An *execution* is a sequence $(q_0, \rho_0), \mathtt{op}_1, (q_1, \rho_1), \dots, \mathtt{op}_k, (q_k, \rho_k)$ such that:

$$(q_0, \rho_0) \xrightarrow{\mathtt{op}_1} (q_1, \rho_1) \cdots \xrightarrow{\mathtt{op}_k} (q_k, \rho_k)$$

A CFA is said to be *unsafe* if there exists an execution from an initial configuration to a bad one. Otherwise the CFA is said to be *safe*. The *reachability problem* for CFA consists in deciding if a CFA is unsafe.

Theorem 2.2. *The reachability problem for CFA is undecidable, even if X is restricted to two variables and the operator $*$ is disallowed in expressions.*

Exercise 2.3. Provide an example of a very simple C program containing a conditional branch $if...then...else$ and an assertion, and a corresponding CFA encoding the problem of deciding if the assertion is always true.