The $\mathbb{K}$ Summarizer

Runtime Verification, Inc.
(written by Xiaohong Chen)

June 20, 2022

The purpose of this document is to give a formal specification of the $\mathbb{K}$ summarizer (`https://github.com/runtimeverification/erc20-verification/tree/master/ksummarize`). We will investigate how the $\mathbb{K}$ summarizer can be used to do semantics-based compilation (SBC) and formal verification.

1 Preliminaries

Throughout this document, we assume that there is an underlying matching logic theory Γ^L that defines the formal semantics of a given programming language L. The provability symbol $\vdash$ in this document should always be understood as

$$\Gamma^L \vdash \varphi$$

i.e., φ is provable with the formal semantics of L.

The following standard notations will be used:

- φ, ψ: an arbitrary pattern.
- t: a term pattern, built from element variables and functional symbols.
- p: a predicate pattern.
- $t \wedge p$: a constraint term.
- $\tau \equiv [\varphi_1/x_1, \ldots, \varphi_n/x_n]$: a substitution.
- $\varphi\tau$ or $\varphi[\varphi_1/x_1, \ldots, \varphi_n/x_n]$: applying the substitution to φ.
- $\lceil\varphi\rceil$: the definedness pattern of φ.
- $\bullet\varphi$: "one-path next" of φ.
- $\circ\varphi$: "all-path next" of φ, defined as $\circ\varphi \equiv \neg\bullet\neg\varphi$.
- STOP: stopped/terminal states; abbreviation for $\circ\bot$.
- $\mathsf{NONSTOP}$: non-stopped states; abbreviation for $\bullet\top$.

- $\circ_s\varphi$: "strongly all-path next" of φ, defined as $\circ_s\varphi \equiv \circ\varphi \wedge \mathsf{NONSTOP}$.
- $\varphi \Rightarrow_1^\exists \varphi'$: "one-step one-path execution"; abbreviation for $\varphi \to \bullet\varphi'$.
- $\varphi \Rightarrow_1^\forall \varphi'$: "one-step all-path execution"; abbreviation for $\varphi \to \circ_s\varphi'$.
- $\varphi \Rightarrow_*^\forall \varphi'$: "all-path execution"; abbreviation for $\varphi \to \mu X \,.\, \varphi \vee \circ X$.
- more to be added . . .

2 Matching and Unification

In this section we formalize matching and unification as proving matching logic theorems. Some definitions and results are from Arusoaie and Lucanu's paper `https://arxiv.org/pdf/1811.02835.pdf`.

2.1 Substitution Patterns

Definition 1. Given a substitution

$$\tau \equiv [\varphi_1/x_1, \ldots, \varphi_n/x_n]$$

we define a corresponding *substitution pattern*

$$\varphi^\tau \equiv (x_1 = \varphi_1) \wedge \cdots \wedge (x_n = \varphi_n)$$

Proposition 2. $\vdash \varphi^\tau \to (\psi = \psi\tau)$.

Proof. This (derived) proof rule is called (Equality Elimination). □

2.2 Matching

Definition 3. Let φ and ψ be two patterns. We say that φ *matches* ψ if

$$\vdash \varphi \to \exists FV(\psi) \,.\, \psi$$

We say that $\{\sigma_1, \ldots, \sigma_n\}$ is a *complete solution* to the matching problem

$$\varphi_1 \lhd_? \psi_1, \ldots, \varphi_m \lhd_? \psi_m$$

if

$$\vdash \left(\bigwedge_{i=1}^{m} \varphi_i \subseteq \psi_i \right) \leftrightarrow \varphi^{\tau_1} \vee \cdots \vee \varphi^{\tau_n}$$

Proposition 4. *For terms t and s, Definition 3 coincides with the classical definition of term matching.*

Proposition 5. *φ matches ψ if and only if*

$$\vdash (\exists FV(\varphi) \,.\, \varphi) \subseteq (\exists FV(\psi) \,.\, \psi)$$

Proof. By Definition 3. □

2.3 Unification

Definition 6. Let φ and ψ be two patterns. We say that φ *unifies* with ψ if

$$\vdash \lceil (\exists FV(\varphi) \,.\, \varphi) \wedge (\exists FV(\psi) \,.\, \psi) \rceil$$

We say that $\{\sigma_1, \ldots, \sigma_n\}$ is a *complete solution* to the unification problem

$$\varphi_1 =_? \psi_1, \ldots, \varphi_m =_? \psi_m$$

if

$$\vdash \left(\bigwedge_{i=1}^{m} \lceil \varphi_i \wedge \psi_i \rceil \right) \leftrightarrow \varphi^{\tau_1} \vee \cdots \vee \varphi^{\tau_n}$$

Proposition 7. *For terms t and s, Definition 6 coincides with the classical definition of term unification.*

2.4 Modulo Theories

Definitions 3 and 6 work with underlying theories, in which case we obtain matching/unification modulo theories.

3 $\mathbb{K}$ Summaries

Definition 8. A $\mathbb{K}$ control-flow graph (abbreviated KCFG) $G = (V, E_r, E_a, E_s)$ is a directed graph with three types of edges where

- the vertex set V is a set of patterns;
- $E_r \subseteq V \times V$ is called the *rewriting relation*;
- $E_a \subseteq V \times V$ is called the *abstracting relation*;
- $E_s \subseteq V \times V$ is called the *splitting relation*.

We write $\varphi \rightsquigarrow_r \psi$ ($\varphi \rightsquigarrow_a \psi$ and $\varphi \rightsquigarrow_s \psi$, resp.) for the three types of edges.

Definition 9. A KCFG $G = (V, E, F)$ is *sound* w.r.t. Γ^L if

1. $\vdash \varphi \Rightarrow_*^\forall \psi$ for all $\varphi \rightsquigarrow_r \psi$;
2. $\vdash \varphi \to \psi$ for all $\varphi \rightsquigarrow_a \psi$;
3. $\vdash \varphi \leftrightarrow \psi_1 \vee \cdots \vee \psi_n$ for all $\varphi, \psi_1, \ldots, \psi_n$ such that $\psi_1, \ldots, \psi_n$ are all the E_s-successors of φ in G.

Intuitively, $\varphi \Rightarrow_*^\forall \psi$ denotes a compilation of many all-path execution steps.