Definition of Safety

Günther Wullaert

May 2022

1 Language

We define our language over four sets of symbols: numerals, symbolic constants, variables, and aggregate names.

1.1 Term and Pools

We inductively define *terms*, *tuples of terms*, and *pools*:

- all numerals and variables are terms,
- $f(\boldsymbol{t})$ is a term if f is a symbolic constant and $\boldsymbol{t}$ is a pool,
- $t_1 \star t_2$ is a term if t_1, t_2 are terms and $\star \in \{+, -, \times, \div, ..\}$,
- $\langle \boldsymbol{t} \rangle$ is a term if $\boldsymbol{t}$ is a pool,
- $t_1, \ldots, t_n$ is a tuple of terms if $n \geq 0$ and t_i is a term,
- $\dot{t_1}; \ldots; \dot{t_n}$ is a pool if $n \geq 1$ and each $\dot{t_i}$ is a tuple of terms.

We omit writing the parenthesis for terms of form $f()$.

We inductively define a term to be *evaluable* if

- it is a numeral, or
- it has form $t_1 \star t_2$ where t_1 and t_2 are evaluable and $\star \in \{+, -, \times, \div\}$.

We then inductively define function *eval* mapping evaluable terms to sets of numerals:

- for numerals t, we let $\mathit{eval}(t) = \{t\}$, and
- for terms of form $t_1 \star t_2$, we let

$$\begin{aligned}
&\mathit{eval}(t_1 + t_2) = \{s_1 + s_2 \mid s_1 \in \mathit{eval}(t_1), s_2 \in \mathit{eval}(t_2)\},\\
&\mathit{eval}(t_1 - t_2) = \{s_1 - s_2 \mid s_1 \in \mathit{eval}(t_1), s_2 \in \mathit{eval}(t_2)\},\\
&\mathit{eval}(t_1 \times t_2) = \{s_1 \times s_2 \mid s_1 \in \mathit{eval}(t_1), s_2 \in \mathit{eval}(t_2)\}, \text{ and}\\
&\mathit{eval}(t_1 \div t_2) = \{s_1 \div s_2 \mid s_1 \in \mathit{eval}(t_1), s_2 \in \mathit{eval}(t_2), s_2 \neq 0\}.
\end{aligned}$$

We say that a term t is *nonzero* if it is evaluable and $0 \notin \mathit{eval}(t)$.

1.2 Atoms and Literals

An *atom* has form $p(\boldsymbol{t})$ where p is a symbolic constant and $\boldsymbol{t}$ is a pool. We omit writing parenthesis for atoms of form $p()$.

A *comparison* has form $t_1 \prec t_2$, where t_1, t_2 are terms and $\prec \in \{<, >, \leq, \geq, =, \neq\}$. Furthermore, we let *negate* be a function to negate relation symbols: $< \mapsto \geq$, $> \mapsto \leq$, $\leq \mapsto >$, $\geq \mapsto <$, $= \mapsto \neq$ and $\neq \mapsto =$.

A *literal* is either an atom or a comparison optionally preceded by the *default negation* symbol $\neg$.

A *conditional literal* has form $l : \dot{l}$, where l is a literal and $\dot{l}$ is a (possibly empty) tuple of literals.

1.3 Aggregates

An *aggregate* has the form

$$\alpha\{\dot{t_1} : \dot{l_1}; \dots; \dot{t_n} : \dot{l_n}\} \prec s \tag{1}$$

where

- $n \geq 0$,
- α is an aggregate name,
- each $\dot{t_i}$ is a tuple of terms,
- each $\dot{l_i}$ is a (possible empty) tuple of literals,
- $\prec \in \{<, >, \leq, \geq, =, \neq\}$, and
- s is a term.

1.4 Rules

A *rule* has form

$$a_1 \vee \dots \vee a_m \leftarrow l_1 \wedge \dots \wedge l_n \tag{2}$$

where $m, n \geq 0$, each a_i is an atom and each l_i is a literal, conditional literal or aggregate.

A *choice rule* has form

$$\{a_1; \dots; a_m\} \leftarrow l_1 \wedge \dots \wedge l_n \tag{3}$$

where $m, n \geq 0$, each a_i is an atom and each l_i is a literal, conditional literal or aggregate.

We refer to the a_i and l_i in rules of form (2) and (3) as *head atoms* and *body literals*, respectively.

2 Safety

In the following, we use function $\mathit{vars}(e)$ to obtain all variables occurring in an expression e. Furthermore, we say that a variable X occurs *globally* in

- a literal l if $X \in \mathit{vars}(l)$,
- a conditional literal $l : \dot{l}$ if $X \in \mathit{vars}(l) \setminus \mathit{vars}(\dot{l})$,
- an aggregate of form (1) if $X \in s$, and
- a rule of form (2) or (3) if it occurs globally in a head atom or body literal.

2.1 Terms

We inductively define function pt for terms, tuples of terms and pools:

- for numerals n, we let $\mathit{pt}(n) = \emptyset$,
- for variables X, we let $\mathit{pt}(X) = \{X\}$,
- for term tuples $\dot{t} = t_1, \dots, t_n$, we let $\mathit{pt}(\dot{t}) = \mathit{pt}(t_1) \cup \dots \cup \mathit{pt}(t_n)$,
- for pools $\boldsymbol{t} = \dot{t_1}; \dots; \dot{t_n}$, we let $\mathit{pt}(\boldsymbol{t}) = \mathit{pt}(\dot{t_1}) \cap \dots \cap \mathit{pt}(\dot{t_n})$,
- for terms of form $f(\boldsymbol{t})$, we let $\mathit{pt}(f(\boldsymbol{t})) = \mathit{pt}(\boldsymbol{t})$,
- for terms of form $t_1 \star t_2$, we let

$$\mathit{pt}(t_1 \star t_2) = \begin{cases} \mathit{pt}(t_2) & t_1 \text{ is evaluable and } \star \in \{+, -\}, \text{ or} \\ & t_1 \text{ is nonzero and } \star = \times, \\ \mathit{pt}(t_1) & t_2 \text{ is evaluable and } \star \in \{+, -\}, \text{ or} \\ & t_2 \text{ is nonzero and } \star = \times, \\ \emptyset & \text{otherwise} \end{cases}$$

We define function dt for terms t as $\mathit{dt}(t) = \mathit{vars}(t) \setminus \mathit{pt}(t)$.

2.2 Body Literals

Next, we define function dep for body literals:

- for an atom a of form $p(\boldsymbol{t})$, we let $\mathit{dep}(a) = \{(\mathit{pt}(\boldsymbol{t}), \emptyset), (\emptyset, \mathit{dt}(\boldsymbol{t}))\}$,
- for a comparison a of form $t_1 \prec t_2$ with $\prec \notin \{=\}$, we let $\mathit{dep}(a) = \{(\emptyset, \mathit{vars}(a))\}$,
- for a comparison a of form $t_1 = t_2$, we let $\mathit{dep}(a) = \{(\mathit{pt}(t_1), \mathit{vars}(t_2)), (\mathit{pt}(t_2), \mathit{vars}(t_1)), (\emptyset, \mathit{dt}(t_1) \cup \mathit{dt}(t_2))\}$,
- for a literal l of form $\neg a$ where a is an atom, we let $\mathit{dep}(l) = \{(\emptyset, \mathit{vars}(l))\}$,

- for a literal l of form $\neg t_1 \prec t_2$, we let $dep(l) = dep(t_1\ negate(\prec)\ t_2)$,
- for conditional literal l, we let $dep(l) = \{(\emptyset, vars(l))\}$,
- for an aggregate l of form (1) with $\prec \notin \{=\}$, we let $dep(l) = \{(\emptyset, vars(l))\}$,
- for an aggregate l of form (1) with $\prec \in \{=\}$, we let $dep(l) = \{(pt(s), vars(\dot{t}_1 : \dot{l}_1; \dots; \dot{t}_n : \dot{l}_n)), (\emptyset, dt(s))\}$.

2.3 Gathering Dependencies

Next, we define function *analyze* to gather dependencies in rules, conditional literals, and aggregate elements:

- for a rule r of form (2) or (3), we let

$$analyze(r) = \bigcup_{1 \le i \le m} \{(\emptyset, vars(a_i))\} \cup \bigcup_{1 \le i \le n} dep(l_i),$$

- for a conditional literal l of form $l_0 : l_1, \dots, l_n$, we let

$$analyze(l) = \{(\emptyset, vars(l_0))\} \cup \bigcup_{1 \le i \le n} dep(l_i),$$

- for an aggregate element e of form $\dot{t} : l_1, \dots, l_n$, we let

$$analyze(e) = \{(\emptyset, vars(\dot{t}))\} \cup \bigcup_{1 \le i \le n} dep(l_i).$$

Given a set PD of pairs of sets of variables and a set of variables V, we let $PD_{|V} = \{(P \cap V, D \cap V) \mid (P, D) \in PD\}$.

2.4 Safety Definition

We define operator C_{PD} parametrized with a set PD of pairs of sets of variables applied to a set V of variables as

$$C_{PD}(V) = \bigcup_{(P,D) \in PD, D \subseteq V} P.$$

Let r be a rule with global variables G. We say that a rule r is *globally safe* if G is the least fixed point of C_{PD} with $PD = analyze(r)_{|G}$. Furthermore, we say that a conditional literal or aggregate element e occurring in rule r is *locally safe* if $L = vars(e) \setminus G$ is the least fixed point of C_{PD} with $PD = analyze(e)_{|L}$.

A rule is *safe* if it is globally safe and all occurrences of conditional literals and aggregate elements in it are locally safe.

3 Other Examples

$$\begin{aligned}
dep(p(X, Y+Y)) &= \{(pt(X, Y+Y), \emptyset), (\emptyset, dt(X, Y+Y))\} \\
&= \{(pt(X) \cup pt(Y+Y), \emptyset), (\emptyset, dt(X) \cup dt(Y+Y))\} \\
&= \{(\{X\} \cup \emptyset, \emptyset), (\emptyset, \emptyset \cup vars(Y+Y))\} \\
&= \{(\{X\}, \emptyset), (\emptyset, \{Y\})\}
\end{aligned}$$

This article was processed using the comments style on July 13, 2022.
There remain 0 comments to be processed.