Definition of Safety

Günther Wullaert

May 2022

1 Language

1.1 Term and Pools

We inductively define terms, tuples of terms, and pools as

- all numerals, symbolic constants, 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 $\star$ is among the symbols +, -, $\times$, / or .. and t_1, t_2 are terms,
- $\langle \boldsymbol{t} \rangle$ is a term, if $\boldsymbol{t}$ is a pool, which can have a possibly empty set of terms,
- $t_1, ..., t_n$ is a tuple of terms, if $n \geq 0$ and t_i is a term,
- $\dot{t_1}; ...; \dot{t_n}$ is a pool, if $n \geq 1$ and each $\dot{t_i}$ is a tuple of terms.

1.2 Constants

We inductively define a term to be *constant* if

- it is a numeral,
- it has form $t \star u$ where t and u are *constant* and $\star$ is among the symbols +, -, $\times$ or /.

1.3 Atoms and Literals

An atom has form $p(\boldsymbol{t})$ where p is a predicate symbol and $\boldsymbol{t}$ is a pool.
A comparison literal has form $t_1 \prec t_2$, where t_1, t_2 are terms and $\prec$ is among the symbols $\leq, \geq, <, >$ or $\neq$.
A conditional literal has form $l : \dot{c}$, where l is a literal and c is a tuple of literals.
A literal is either an

- atom,
- comparison literal or

- conditional literal,

which can be preceded by not.

1.4 Aggregates

An aggregate has one of the forms

$$\alpha\{\dot{t_1} : \dot{L_1}; \ldots ; \dot{t_n} : \dot{L_n}\} \prec_1 s_1 \tag{1}$$

$$s_1 \prec_1 \alpha\{\dot{t_1} : \dot{L_1}; \ldots ; \dot{t_n} : \dot{L_n}\} \prec_2 s_2 \tag{2}$$

$(n \leq 0)$, where

- α is an aggregate name,
- each $\dot{t_i}$ is a tuple of terms,
- each $\dot{L_i}$ is a tuple of comparison literals and atoms,
- each $\prec_1, \prec_2$ is among the symbols $\leq, \geq, <, >, =$ or $\neq$,
- each s_1, s_2 is a term.

1.5 Rules

A rule r has the form

$$H_1 \vee ... \vee H_m \leftarrow B_1 \wedge ... \wedge B_n \tag{3}$$

$(m, n \geq 0)$, where each H_i is a literal and each B_j is a literal or an aggregate. $H_1 \vee ... \vee H_m$ is called the head. $B_1 \wedge ... \wedge B_n$ is called the body.

2 Safety

The $vars(e)$ function returns all variables for an expression e.
For Example:

$$vars(a(X) = b(Y)) = \{X, Y\}$$

The $eval(c)$ function takes a constant term and returns the arithmetic evaluation of that term. If this function for a *constant* returns a set s, where $0 \notin s$ then this result is called *nonzero*.

$$\begin{aligned}
eval(c) &= \{c\} \\
eval(a + b) &= \{x + y \qquad | \; x \in eval(a), y \in eval(b)\} \\
eval(a - b) &= \{x - y \qquad | \; x \in eval(a), y \in eval(b)\} \\
eval(a * b) &= \{x * y \qquad | \; x \in eval(a), y \in eval(b)\} \\
eval(a/b) &= \{x/y \qquad | \; x \in eval(a), y \in eval(b), y \neq 0\}
\end{aligned}$$

2.1 Terms

2.1.1 Constants

For any numeral n and symbolic constant f

$$pt(n) = pt(f) = dt(n) = dt(f) = \emptyset$$

2.1.2 Variables

For any variable X

$$\begin{aligned} pt(X) &= \{X\} \\ dt(X) &= \emptyset \end{aligned}$$

2.1.3 Tuples

For any tuple of terms $t_1, ..., t_n$

$$\begin{aligned} pt(t_1, ..., t_n) &= pt(t_1) \cup \dots \cup pt(t_n) \\ dt(t_1, ..., t_n) &= dt(t_1) \cup \dots \cup dt(t_n) \end{aligned}$$

2.1.4 Pools

For any pool of terms $\dot{t}_1; ...; \dot{t}_n$

$$\begin{aligned} pt(\dot{t}_1; ...; \dot{t}_n) &= pt(\dot{t}_1) \cap \dots \cap pt(\dot{t}_n) \\ dt(\dot{t}_1; ...; \dot{t}_n) &= dt(\dot{t}_1) \cup \dots \cup dt(\dot{t}_n) \end{aligned}$$

2.1.5 Functions

For a term of form $f(\boldsymbol{t})$, where f is a function and $\boldsymbol{t}$ a pool

$$\begin{aligned} pt(f(\boldsymbol{t})) &= pt(\boldsymbol{t}) \\ dt(f(\boldsymbol{t})) &= dt(\boldsymbol{t}) \end{aligned}$$

2.1.6 Arithmetics

For a term of form $t \times c$, where t is a term and c a constant and $eval(c) = 0$

$$\begin{aligned} pt(t \times c) &= \emptyset \\ dt(t \times c) &= \emptyset \end{aligned}$$

For a term of form $a \star b$, where a, b are terms and one of them is a *constant* and the *constant* is *nonzero* and $\star$ is among the symbols +, - or × or a and b are both *constant* and $\star$ is among the symbols +, -, ×, / or ..

$$\begin{aligned} pt(a \star b) &= pt(a) \cup pt(b) \\ dt(a \star b) &= dt(a) \cup dt(b) \end{aligned}$$

Otherwise for a term of form $a \star b$

$$\begin{aligned} pt(a \star b) &= \emptyset \\ pt(a \star b) &= vars(a \star b) \end{aligned}$$

For a term of form $-t$, where t is a term

$$\begin{aligned} pt(-t) &= pt(t) \\ dt(-t) &= dt(t) \end{aligned}$$

2.2 Atoms and Literals

2.2.1 Atoms and Literals

For an literal of form *not* a, where a is an atom

$$dep(not\ a) = \{(\emptyset, vars(a))\}$$

For an atom of form $p(\boldsymbol{t})$, where $\boldsymbol{t}$ is a pool

$$dep(p(\boldsymbol{t})) = \{(pt(\boldsymbol{t}), \emptyset), (\emptyset, dt(\boldsymbol{t}))\}$$

2.2.2 Comparison Literals

For an comparison literal of form $a \prec b$, where a and b are terms and $\prec$ is among the symbols $\leq, \geq, <, >, \neq$

$$dep(a \prec b) = \{(\emptyset, vars(a \prec b))\}$$

For an comparison literal of form $a = b$, where a and b are terms

$$dep(a = b) = \{(pt(a), vars(b)), (pt(b), vars(a)), (\emptyset, dt(a) \cup dt(b))\}$$

For an literal of form *not* $a \prec b$, where a and b are terms

$$\begin{aligned} dep(not\ a = b) &= dep(a \neq b) \\ dep(not\ a \neq b) &= dep(a = b) \\ dep(not\ a \leq b) &= dep(a > b) \\ dep(not\ a > b) &= dep(a \leq b) \\ dep(not\ a \geq b) &= dep(a < b) \\ dep(not\ a < b) &= dep(a \geq b) \end{aligned}$$

2.2.3 Conditional Literals

For a conditional literal of form $\dot{t} : \dot{c}$, where t and c are tuples

$$dep(\dot{t} : \dot{c}) = \emptyset$$

2.3 Aggregates

We define $elem(a)$ to return a set of elements in an aggregate a as

$$elem(a) = \{\dot{t_1} : \dot{L_1}, \ldots, \dot{t_n} : \dot{L_n}\}$$

For an aggregate disjunction of form $\dot{t_1} : \dot{L_1}$, where t_1 is a tuple of terms and $\dot{L_1}$ is a tuple of comparison literals and atoms

$$dep(\dot{t_1} : \dot{L_1}) = \{(\emptyset, vars(\dot{t_1}))\} \cup dep(\dot{L_1})$$

For an aggregate a, where $\prec_1$ is $=$

$$dep(a) = \{(vars(s_1), \emptyset)\}$$

Otherwise

$$dep(a) = \emptyset$$

2.4 Rule

For a rule r in the form of (3) the following holds:

$$dep(r) = \{(\emptyset, vars(H_1 \vee ... \vee H_k))\} \cup dep(B_1) \cup ... \cup dep(B_m)$$

2.5 Safety Definition

For a aggragate a occuring in a rule where G is the set of variables occuring globally in it. We define operator $C_{e,G}$ for an element e of an aggregate a applied to a set of variables V as

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

a is safe if for each element $e \in elem(a)$, $vars(e)$ is the least fixed point of $C_{e,G}$ We define operator C_r for a rule r applied to a set of variables V as

$$C_r(V) = \bigcup_{(P,D) \in dep(r), D \subseteq V} P.$$

A rule r is safe if $vars(r)$ is the least fixed point of C_r and each aggregate is safe.

3 Other Examples

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