Definition of Safety

Günther Wullaert

May 2022

1 Language

1.1 Term and Pools

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

- all numerals, symbolic constants, and variables are terms,[1]
- $f(\boldsymbol{t})$ 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,
- $t_1, \dots, t_n$ is a tuple of terms, if $n \geq 0$ and t_i is a term,
- $\dot{t_1}; \dots; \dot{t_n}$ is a pool, if $n \geq 1$ and each $\dot{t_i}$ is a tuple of terms.[2]

[1] R: Define those things.

[2] R: Why not empty?

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 inductively define function *eval* to map 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 c is *nonzero* if it is evaluable and $0 \notin \mathit{eval}(c)$.

1.2 Atoms and Literals

An *atom* has form $p(\boldsymbol{t})$ where p is a predicate symbol and $\boldsymbol{t}$ is a pool.[3]

[3] R: Define predicate symbols.

A *comparison* 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 *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

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

where

- $n \geq 0$,
- α is an aggregate name,
- each $\dot{t_i}$ is a tuple of terms,
- each $\dot{l_i}$ is a tuple of literals,
- $\prec$ is among the symbols $\leq$, $\geq$, $<$, $>$, $=$ or $\neq$, and
- s is a term.

1.4 Rules

A *rule* has form

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

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

A *choice rule* has form

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

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

We refer to the atoms a_i and literals 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. 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 and symbolic constants f, we let $\mathit{pt}(n) = \mathit{pt}(f) = \emptyset$,
- 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 Atoms and Literals

2.2.1 Atoms and Literals

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

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

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

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

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 is a tuple of terms and c is a tuple of comparison literals and atoms, where G is the set of variables occuring globally in it.

$$dep(\dot{t} : \dot{c}) = \{(\emptyset, G \cup vars(s))\}$$

For an conditional literal of form $\dot{t} : \dot{c}$, where t is a tuple of terms and c is a tuple of comparison literals and atoms in a local context

$$dep_l(\dot{t} : \dot{c}) = \{(\emptyset, vars(\dot{t}))\} \cup \bigcup_{e \in \dot{c}} dep(e)$$

2.3 Aggregates

For a aggragate a occuring in a rule, where G is the set of variables occuring globally in it. We define $elem(a)$ to return a set of elements in a as

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

For an aggregate element 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

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

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

$$dep(a) = \{(pt(s), G), (\emptyset, dt(s) \cup G)\}$$

Otherwise

$$dep(a) = \{(\emptyset, G \cup vars(s))\}$$

2.4 Rule

For a rule r in the form of (2) the following holds:

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

2.5 Choice Rule

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

$$dep(r) = \{(\emptyset, vars(e_1 \vee ... \vee e_k))\} \cup dep(B_1) \cup ... \cup dep(B_m)$$

2.6 Safety Definition

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 globally safe if $vars(r)$ is the least fixed point of C_r and each aggregate and conditional literal is locally safe.
We define operator $C_{e,G}$ for an element e of an aggregate a applied to a set of variables V as

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

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

$$C_{l,G}(V) = G \cup \bigcup_{(P,D) \in dep_l(l), D \subseteq V} P.$$

l is locally safe if $vars(l)$ is the least fixed point of $C_{l,G}$

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 12, 2022.
There remain 3 comments to be processed.