To Be or Not To Be Created

Abstract Object Creation in Dynamic Logic

Wolfgang Ahrendt[1] Frank S. de Boer[2] Immo Grabe[3]

[1]Chalmers University, Göteborg, Sweden

[2]CWI, Amsterdam, The Netherlands

[3]Christian-Albrechts-University Kiel, Germany

Types Club, 22/04/2021

A Simple Object-Oriented While Language

- only one class: Object
- 3 types: Object, Integer, Boolean
- no methods
- variables (e.g. u, v, w) distinct from fields (e.g. x, y, z)

statements:

$$
\begin{aligned}
s \quad ::= \quad & \text{while } e \text{ do } s \text{ od} \mid s_1; s_2 \mid \\
& u := e \mid e_1.x := e_2 \mid u := \text{new}
\end{aligned}
$$

expressions:

$$e \quad ::= \quad u \mid e.x \mid \text{null} \mid e_1 = e_2 \mid op(e_1, ..., e_n)$$

to separate issues object creation and aliasing:

- no native statement $e.x := \text{new}$
- can be simulated by $u := \text{new}; e.x := u$ (u fresh)

Semantics

informal in this talk

- $[\![u := \text{new}]\!]_\sigma$: create new object and assign it to u
- $[\![e]\!]_\sigma \in$ set of objects existing in σ
- $[\![\forall o.\phi]\!]_\sigma$: ϕ holds for all objects existing in σ
- $[\![\exists o.\phi]\!]_\sigma$: ϕ holds for some object existing in σ

e, o of type Object

examples:

$\forall o.\langle u := \text{new}\rangle\neg(u = o)$ true in all states

$\langle u := \text{new}\rangle\forall o.\neg(u = o)$ false in all states

Update Application: Restricted Standard Cases

The standard rules for *quantifiers* and *equality* are restricted to non-creating updates $\mathcal{U}_{nc}$ of the forms '$u := e$', '$e_1.x := e_2$'.
('$u :=$ new' excluded from these rules.)

$$\frac{\forall l.\, \{\mathcal{U}_{nc}\}\phi \rightsquigarrow \phi'}{\{\mathcal{U}_{nc}\}(\forall l.\, \phi) \rightsquigarrow \phi'}$$

$$\frac{\exists l.\, \{\mathcal{U}_{nc}\}\phi \rightsquigarrow \phi'}{\{\mathcal{U}_{nc}\}(\exists l.\, \phi) \rightsquigarrow \phi'}$$

$$\frac{\{\mathcal{U}_{nc}\}e_1 = \{\mathcal{U}_{nc}\}e_2 \rightsquigarrow e'}{\{\mathcal{U}_{nc}\}(e_1 = e_2) \rightsquigarrow e'}$$

Object Creating Updates: the Issue

note:

- '$\{\mathcal{U}\}\phi$' is the (explicit) weakest precondition $wp(\mathcal{U}, \phi)$

problem:

- result of $\{u := new\}\phi$, i.e., $wp(\{u := new\}, \phi)$, cannot talk about the new object because it does not exist in pre-state
- in particular: $\{u := new\}u \rightsquigarrow$?

basic approach:

- totally avoid '$\{u := new\}u$'
- observation: the only operations on objects are
  - de-referencing fields
  - test for equality
  - quantification
- in all cases, wp computation can employ meta knowledge

Object Creating Update Application: Field Access

$$\frac{(\{u := new\}e).x \rightsquigarrow e'}{\{u := new\}(e.x) \rightsquigarrow e'}$$

$$e \not\equiv u$$

$$\{u := new\}u.x \rightsquigarrow init_{T(x)}$$

$$init_{T(x)} \equiv \text{null} \mid 0 \mid \text{false}$$

Object Creating Update Application: Equality

$$\frac{(\{u := new\}e_1) = (\{u := new\}e_2) \rightsquigarrow e'}{\{u := new\}(e_1 = e_2) \rightsquigarrow e'}$$

$$e_1 \not\equiv u, \quad e_2 \not\equiv u$$

$$\{u := new\}(u = e) \rightsquigarrow \text{false}$$

$$e \not\equiv u$$

$$\{u := new\}(u = u) \rightsquigarrow \text{true}$$

Object Creating Update Application: Quantifiers

$$\frac{(\{u := new\}\phi(u)) \wedge \forall o.(\{u := new\}\phi(o)) \rightsquigarrow \phi'}{\{u := new\}\forall o.\phi(o) \rightsquigarrow \phi'}$$

$$\frac{(\{u := new\}\phi(u)) \vee \exists o.(\{u := new\}\phi(o)) \rightsquigarrow \phi'}{\{u := new\}\exists o.\phi(o) \rightsquigarrow \phi'}$$

Abstract Object Creation Proof

$$
\textit{allRight}\;\dfrac{\textit{assignVar}\;\dfrac{\leadsto\;\dfrac{\textit{notRight}\;\dfrac{\textit{closeFalse}\;\dfrac{*}{\mathsf{false} \Longrightarrow}}{\Longrightarrow \neg\, \mathsf{false}}}{\Longrightarrow \{u := \mathsf{new}\}\neg(u = c)}}{\Longrightarrow \langle u := \mathsf{new}\rangle\neg(u = c)}}{\Longrightarrow \forall o. \langle u := \mathsf{new}\rangle\neg(u = o)}
$$

Object Activation Proof

$$
\textit{allRight}\ \dfrac{\textit{impRight}\ \dfrac{\textit{createObj}\ \dfrac{\textit{applyUpd}\ \dfrac{\textit{notRight}\ \dfrac{\textit{inReachableState}\ \dfrac{\textit{allLeft}\ \dfrac{(\approx 2\ \textit{rules})\ \dfrac{\textit{notLeft}\ \dfrac{\textit{equality}\ \dfrac{\textit{close}\ \dfrac{*}{c.\mathrm{cre},\ \mathrm{obj}(\mathrm{next}) = c \Longrightarrow c.\mathrm{cre}}}{c.\mathrm{cre},\ \mathrm{obj}(\mathrm{next}) = c \Longrightarrow \mathrm{obj}(\mathrm{next}).\mathrm{cre}}}{\neg\mathrm{obj}(\mathrm{next}).\mathrm{cre},\ c.\mathrm{cre},\ \mathrm{obj}(\mathrm{next}) = c \Longrightarrow}}{(\mathrm{obj}(\mathrm{next}).\mathrm{cre} \leftrightarrow \mathrm{next} < \mathrm{next}),\ c.\mathrm{cre},\ \mathrm{obj}(\mathrm{next}) = c \Longrightarrow}}{\forall n.(\mathrm{obj}(n).\mathrm{cre} \leftrightarrow n < \mathrm{next}),\ c.\mathrm{cre},\ \mathrm{obj}(\mathrm{next}) = c \Longrightarrow}}{c.\mathrm{cre},\ \mathrm{obj}(\mathrm{next}) = c \Longrightarrow}}{c.\mathrm{cre} \Longrightarrow \neg(\mathrm{obj}(\mathrm{next}) = c)}}{c.\mathrm{cre} \Longrightarrow \{u := \mathrm{obj}(\mathrm{next});\ u.\mathrm{cre} := \mathrm{true};\ \mathrm{next} := \mathrm{next}+1\}\neg(u = c)}}{c.\mathrm{cre} \Longrightarrow \langle u := \mathrm{new}\rangle\neg(u = c)}}{\Longrightarrow c.\mathrm{cre} \rightarrow \langle u := \mathrm{new}\rangle\neg(u = c)}}{\Longrightarrow \forall o.\,(o.\mathrm{cre} \rightarrow \langle u := \mathrm{new}\rangle\neg(u = o))}
$$