Formalization of a Framework for the Sound Automation of Magic Wands

Thibault Dardinier

May 27, 2022

Abstract

The magic wand $-\!\!*$ (also called separating implication) is a separation logic [4] connective commonly used to specify properties of partial data structures, for instance during iterative traversals. A *footprint* of a magic wand formula $A -\!\!* B$ is a state that, combined with any state in which A holds, yields a state in which B holds. The key challenge of proving a magic wand (also called *packaging* a wand) is to find such a footprint. Existing package algorithms either have a high annotation overhead or are unsound.

In this entry, we formally define a framework for the sound automation of magic wands, described in a paper at CAV 2022 [2], and prove that it is sound and complete. This framework, called the *package logic*, precisely characterises a wide design space of possible package algorithms applicable to a large class of separation logics.

Contents

1 Separation Algebra

In this section, we formalize the concept of a separation algebra [1, 3], on which our package logic is based.

theory *SepAlgebra*
imports *Main*
begin

type-synonym *'a property* = *'a* ⇒ *bool*

locale *sep-algebra* =

fixes *plus* :: *'a* ⇒ *'a* ⇒ *'a option* (**infixl** ⊕ *63*)

fixes *core* :: *'a* ⇒ *'a* (|-|)

assumes *commutative*: *a* ⊕ *b* = *b* ⊕ *a*
and *asso1*: *a* ⊕ *b* = *Some ab* ∧ *b* ⊕ *c* = *Some bc* ⟹ *ab* ⊕ *c* = *a* ⊕ *bc*
and *asso2*: *a* ⊕ *b* = *Some ab* ∧ *b* ⊕ *c* = *None* ⟹ *ab* ⊕ *c* = *None*

and *core-is-smaller*: *Some x* = *x* ⊕ |*x*|
and *core-is-pure*: *Some* |*x*| = |*x*| ⊕ |*x*|
and *core-max*: *Some x* = *x* ⊕ *c* ⟹ (∃ *r*. *Some* |*x*| = *c* ⊕ *r*)
and *core-sum*: *Some c* = *a* ⊕ *b* ⟹ *Some* |*c*| = |*a*| ⊕ |*b*|

and *positivity*: *a* ⊕ *b* = *Some c* ⟹ *Some c* = *c* ⊕ *c* ⟹ *Some a* = *a* ⊕ *a*
and *cancellative*: *Some a* = *b* ⊕ *x* ⟹ *Some a* = *b* ⊕ *y* ⟹ |*x*| = |*y*| ⟹ *x* = *y*

begin

lemma *asso3*:
assumes *a* ⊕ *b* = *None*
and *b* ⊕ *c* = *Some bc*
shows *a* ⊕ *bc* = *None*
⟨*proof*⟩

1.1 Definitions

definition *defined* :: *'a* ⇒ *'a* ⇒ *bool* (**infixl** ## *62*) **where**
a ## *b* ⟷ *a* ⊕ *b* ≠ *None*

definition *greater* :: *'a* ⇒ *'a* ⇒ *bool* (**infixl** ≽ *50*) **where**
a ≽ *b* ⟷ (∃ *c*. *Some a* = *b* ⊕ *c*)

definition *pure* :: *'a* ⇒ *bool* **where**
pure a ⟷ *Some a* = *a* ⊕ *a*

definition *minus* :: *'a* ⇒ *'a* ⇒ *'a* (**infixl** ⊖ *63*)
where $b \ominus a = (THE\text{-}default\ b\ (\lambda x.\ Some\ b = a \oplus x \wedge x \succeq |b|\ ))$

definition *add-set* :: *'a set* ⇒ *'a set* ⇒ *'a set* (**infixl** ⊗ *60*) **where**
$A \otimes B = \{\ \varphi \mid \varphi\ a\ b.\ a \in A \wedge b \in B \wedge Some\ \varphi = a \oplus b\ \}$

definition *greater-set* :: *'a set* ⇒ *'a set* ⇒ *bool* (**infixl** ≫ *50*) **where**
$A \gg B \longleftrightarrow (\forall\, a \in A.\ \exists\, b \in B.\ a \succeq b)$

definition *up-closed* :: *'a set* ⇒ *bool* **where**
$up\text{-}closed\ A \longleftrightarrow (\forall\, \varphi'.\ (\exists\, \varphi \in A.\ \varphi' \succeq \varphi) \longrightarrow \varphi' \in A)$

definition *equiv* :: *'a set* ⇒ *'a set* ⇒ *bool* (**infixl** ∼ *40*) **where**
$A \sim B \longleftrightarrow A \gg B \wedge B \gg A$

definition *setify* :: *'a property* ⇒ (*'a set* ⇒ *bool*) **where**
$setify\ P\ A \longleftrightarrow (\forall\, x \in A.\ P\ x)$

definition *mono-prop* :: *'a property* ⇒ *bool* **where**
$mono\text{-}prop\ P \longleftrightarrow (\forall\, x\ y.\ y \succeq x \wedge P\ x \longrightarrow P\ y)$

definition *under* :: *'a set* ⇒ *'a* ⇒ *'a set* **where**
$under\ A\ \omega = \{\ \omega' \mid \omega'.\ \omega' \in A \wedge \omega \succeq \omega'\}$

definition *max-projection-prop* :: (*'a* ⇒ *bool*) ⇒ (*'a* ⇒ *'a*) ⇒ *bool* **where**
$max\text{-}projection\text{-}prop\ P\ f \longleftrightarrow (\forall\, x.\ x \succeq f\ x \wedge P\ (f\ x) \wedge$
$(\forall\, p.\ P\ p \wedge x \succeq p \longrightarrow f\ x \succeq p))$

inductive *multi-plus* :: *'a list* ⇒ *'a* ⇒ *bool* **where**
MPSingle: $multi\text{-}plus\ [a]\ a$
| *MPConcat*: $[\![\ length\ la > 0\ ;\ length\ lb > 0\ ;\ multi\text{-}plus\ la\ a\ ;\ multi\text{-}plus\ lb\ b\ ;$
$Some\ \omega = a \oplus b\ ]\!] \Longrightarrow multi\text{-}plus\ (la\ @\ lb)\ \ \omega$

fun *splus* :: *'a option* ⇒ *'a option* ⇒ *'a option* **where**
$splus\ None\ \text{-} = None$
| $splus\ \text{-}\ None = None$
| $splus\ (Some\ a)\ (Some\ b) = a \oplus b$

1.2 First lemmata

lemma *greater-equiv*:
$a \succeq b \longleftrightarrow (\exists\, c.\ Some\ a = c \oplus b)$
⟨*proof*⟩

lemma *smaller-compatible*:
assumes $a' \#\# b$
and $a' \succeq a$
shows $a \#\# b$

⟨*proof*⟩

lemma *bigger-sum-smaller*:
assumes *Some* $c = a \oplus b$
and $a \succeq a'$
shows $\exists\, b'.\ b' \succeq b \wedge$ *Some* $c = a' \oplus b'$
⟨*proof*⟩

1.2.1 splus

lemma *splus-develop*:
assumes *Some* $a = b \oplus c$
shows $a \oplus d =$ *splus* (*splus* (*Some* b) (*Some* c)) (*Some* d)
⟨*proof*⟩

lemma *splus-comm*:
splus $a\ b =$ *splus* $b\ a$
⟨*proof*⟩

lemma *splus-asso*:
splus (*splus* $a\ b$) $c =$ *splus* a (*splus* $b\ c$)
⟨*proof*⟩

1.2.2 Pure

lemma *pure-stable*:
assumes *pure* a
and *pure* b
and *Some* $c = a \oplus b$
shows *pure* c
⟨*proof*⟩

lemma *pure-smaller*:
assumes *pure* a
and $a \succeq b$
shows *pure* b
⟨*proof*⟩

1.3 Succ is an order

lemma *succ-antisym*:
assumes $a \succeq b$
and $b \succeq a$
shows $a = b$
⟨*proof*⟩

lemma *succ-trans*:
assumes $a \succeq b$
and $b \succeq c$

shows $a \succeq c$
⟨*proof*⟩

lemma *succ-refl*:
$a \succeq a$
⟨*proof*⟩

1.4 Core (pure) and stabilize (stable)

lemma *max-projection-propI*:
assumes $\bigwedge x.\ x \succeq f\ x$
and $\bigwedge x.\ P\ (f\ x)$
and $\bigwedge x\ p.\ P\ p \wedge x \succeq p \Longrightarrow f\ x \succeq p$
shows *max-projection-prop P f*
⟨*proof*⟩

lemma *max-projection-propE*:
assumes *max-projection-prop P f*
shows $\bigwedge x.\ x \succeq f\ x$
and $\bigwedge x.\ P\ (f\ x)$
and $\bigwedge x\ p.\ P\ p \wedge x \succeq p \Longrightarrow f\ x \succeq p$
⟨*proof*⟩

lemma *max-projection-prop-pure-core*:
max-projection-prop pure core
⟨*proof*⟩

lemma *mpp-smaller*:
assumes *max-projection-prop P f*
shows $x \succeq f\ x$
⟨*proof*⟩

lemma *mpp-compatible*:
assumes *max-projection-prop P f*
and $a \#\# b$
shows $f\ a \#\# f\ b$
⟨*proof*⟩

lemma *mpp-prop*:
assumes *max-projection-prop P f*
shows $P\ (f\ x)$
⟨*proof*⟩

lemma *mppI*:
assumes *max-projection-prop P f*
and $a \succeq x$

and $P\ x$
and $x \succeq f\ a$
shows $x = f\ a$
⟨*proof*⟩

lemma *mpp-invo*:
assumes *max-projection-prop* $P\ f$
shows $f\ (f\ x) = f\ x$
⟨*proof*⟩

lemma *mpp-mono*:
assumes *max-projection-prop* $P\ f$
and $a \succeq b$
shows $f\ a \succeq f\ b$
⟨*proof*⟩

1.5 Subtraction

lemma *addition-bigger*:
assumes $a' \succeq a$
and *Some* $x' = a' \oplus b$
and *Some* $x = a \oplus b$
shows $x' \succeq x$
⟨*proof*⟩

lemma *smaller-than-core*:
assumes $y \succeq x$
and *Some* $z = x \oplus |y|$
shows $|z| = |y|$
⟨*proof*⟩

lemma *extract-core*:
assumes *Some* $b = a \oplus x \wedge x \succeq |b|$
shows $|x| = |b|$
⟨*proof*⟩

lemma *minus-unique*:
assumes *Some* $b = a \oplus x \wedge x \succeq |b|$
and *Some* $b = a \oplus y \wedge y \succeq |b|$
shows $x = y$
⟨*proof*⟩

lemma *minus-exists*:
assumes $b \succeq a$
shows $\exists x.$ *Some* $b = a \oplus x \wedge x \succeq |b|$
⟨*proof*⟩

lemma *minus-equiv-def*:
assumes $b \succeq a$
shows *Some* $b = a \oplus (b \ominus a) \wedge (b \ominus a) \succeq |b|$
⟨*proof*⟩

lemma *minus-default*:
assumes $\neg\ b \succeq a$
shows $b \ominus a = b$
⟨*proof*⟩

lemma *minusI*:
assumes *Some* $b = a \oplus x$
and $x \succeq |b|$
shows $x = b \ominus a$
⟨*proof*⟩

lemma *minus-core*:
$|a \ominus b| = |a|$
⟨*proof*⟩

lemma *minus-core-weaker*:
$|a \ominus b| = |a| \ominus |b|$
⟨*proof*⟩

lemma *minus-equiv-def-any-elem*:
assumes *Some* $x = a \oplus b$
shows *Some* $(x \ominus a) = b \oplus |x|$
⟨*proof*⟩

lemma *minus-bigger*:
assumes *Some* $x = a \oplus b$
shows $x \ominus a \succeq b$
⟨*proof*⟩

lemma *minus-smaller*:
assumes $x \succeq a$
shows $x \succeq x \ominus a$
⟨*proof*⟩

lemma *minus-sum*:
assumes *Some* $a = b \oplus c$
and $x \succeq a$
shows $x \ominus a = (x \ominus b) \ominus c$
⟨*proof*⟩

lemma *smaller-compatible-core*:
assumes $y \succeq x$
shows $x \#\# |y|$

⟨*proof*⟩

lemma *smaller-pure-sum-smaller*:
assumes $y \succeq a$
and $y \succeq b$
and *Some* $x = a \oplus b$
and *pure* b
shows $y \succeq x$
⟨*proof*⟩

lemma *greater-minus-trans*:
assumes $y \succeq x$
and $x \succeq a$
shows $y \ominus a \succeq x \ominus a$
⟨*proof*⟩

lemma *minus-and-plus*:
assumes *Some* $\omega' = \omega \oplus r$
and $\omega \succeq a$
shows *Some* $(\omega' \ominus a) = (\omega \ominus a) \oplus r$
⟨*proof*⟩

1.6 Lifting the algebra to sets of states

lemma *add-set-commm*:
$A \otimes B = B \otimes A$
⟨*proof*⟩

lemma *x-elem-set-product*:
$x \in A \otimes B \longleftrightarrow (\exists\, a\ b.\ a \in A \wedge b \in B \wedge \mathit{Some}\ x = a \oplus b)$
⟨*proof*⟩

lemma *x-elem-set-product-splus*:
$x \in A \otimes B \longleftrightarrow (\exists\, a\ b.\ a \in A \wedge b \in B \wedge \mathit{Some}\ x = \mathit{splus}\ (\mathit{Some}\ a)\ (\mathit{Some}\ b))$
⟨*proof*⟩

lemma *add-set-asso*:
$(A \otimes B) \otimes C = A \otimes (B \otimes C)$ (**is** *?A* = *?B*)
⟨*proof*⟩

lemma *up-closedI*:
assumes $\bigwedge \varphi'\ \varphi.\ (\varphi' :: 'a) \succeq \varphi \wedge \varphi \in A \Longrightarrow \varphi' \in A$
shows *up-closed* A
⟨*proof*⟩

lemma *up-closed-plus-UNIV*:

up-closed $(A \otimes UNIV)$
⟨*proof*⟩

lemma *succ-set-trans*:
assumes $A \gg B$
and $B \gg C$
shows $A \gg C$
⟨*proof*⟩

lemma *greater-setI*:
assumes $\bigwedge a.\ a \in A \Longrightarrow (\exists\, b \in B.\ a \succeq b)$
shows $A \gg B$
⟨*proof*⟩

lemma *bigger-set*:
assumes $A' \gg A$
shows $A' \otimes B \gg A \otimes B$
⟨*proof*⟩

lemma *bigger-singleton*:
assumes $\varphi' \succeq \varphi$
shows $\{\varphi'\} \gg \{\varphi\}$
⟨*proof*⟩

lemma *add-set-elem*:
$\varphi \in A \otimes B \longleftrightarrow (\exists\, a\ b.\ Some\ \varphi = a \oplus b \wedge a \in A \wedge b \in B)$
⟨*proof*⟩

lemma *up-closed-sum*:
assumes *up-closed A*
shows *up-closed* $(A \otimes B)$
⟨*proof*⟩

lemma *up-closed-bigger-subset*:
assumes *up-closed B*
and $A \gg B$
shows $A \subseteq B$
⟨*proof*⟩

lemma *up-close-equiv*:
assumes *up-closed A*
and *up-closed B*
shows $A \sim B \longleftrightarrow A = B$
⟨*proof*⟩

lemma *equiv-stable-sum*:
assumes $A \sim B$
shows $A \otimes C \sim B \otimes C$

⟨*proof*⟩

lemma *equiv-up-closed-subset*:
assumes *up-closed A*
and *equiv B C*
shows $B \subseteq A \longleftrightarrow C \subseteq A$ (**is** *?B* $\longleftrightarrow$ *?C*)
⟨*proof*⟩

lemma *mono-propI*:
assumes $\bigwedge x\ y.\ y \succeq x \wedge P\ x \Longrightarrow P\ y$
shows *mono-prop P*
⟨*proof*⟩

lemma *mono-prop-set*:
assumes $A \gg B$
and *setify P B*
and *mono-prop P*
shows *setify P A*
⟨*proof*⟩

lemma *mono-prop-set-equiv*:
assumes *mono-prop P*
and *equiv A B*
shows *setify P A* $\longleftrightarrow$ *setify P B*
⟨*proof*⟩

lemma *setify-sum*:
setify P $(A \otimes B) \longleftrightarrow (\forall x \in A.$ *setify P* $(\{x\} \otimes B))$ (**is** *?A* $\longleftrightarrow$ *?B*)
⟨*proof*⟩

lemma *setify-sum-image*:
setify P $((Set.image\ f\ A) \otimes B) \longleftrightarrow (\forall x \in A.$ *setify P* $(\{f\ x\} \otimes B))$
⟨*proof*⟩

lemma *equivI*:
assumes $A \gg B$
and $B \gg A$
shows *equiv A B*
⟨*proof*⟩

lemma *sub-bigger*:
assumes $A \subseteq B$
shows $A \gg B$
⟨*proof*⟩

lemma *larger-set-refl*:
$A \gg A$
⟨*proof*⟩

definition *upper-closure* **where**
upper-closure A = { φ' |φ' φ. $\varphi' \succeq \varphi \wedge \varphi \in A$ }

lemma *upper-closure-up-closed*:
up-closed (*upper-closure A*)
⟨*proof*⟩

1.7 Addition of more than two states

lemma *multi-decompose*:
assumes *multi-plus l ω*
shows *length l* ≥ *2* ⟹ (∃ *a b la lb*. *l* = *la* @ *lb* ∧ *length la* > *0* ∧ *length lb* > *0* ∧ *multi-plus la a* ∧ *multi-plus lb b* ∧ *Some ω* = *a* ⊕ *b*)
⟨*proof*⟩

lemma *multi-take-drop*:
assumes *multi-plus l ω*
and *length l* ≥ *2*
shows ∃ *n a b*. *n* > *0* ∧ *n* < *length l* ∧ *multi-plus* (*take n l*) *a* ∧ *multi-plus* (*drop n l*) *b* ∧ *Some ω* = *a* ⊕ *b*
⟨*proof*⟩

lemma *multi-plus-single*:
assumes *multi-plus* [*v*] *a*
shows *a* = *v*
⟨*proof*⟩

lemma *multi-plus-two*:
assumes *length l* ≥ *2*
shows *multi-plus l ω* ⟷ (∃ *a b la lb*. *l* = (*la* @ *lb*) ∧ *length la* > *0* ∧ *length lb* > *0* ∧ *multi-plus la a* ∧ *multi-plus lb b* ∧ *Some ω* = *a* ⊕ *b*) (**is** *?A* ⟷ *?B*)
⟨*proof*⟩

lemma *multi-plus-head-tail*:
length l ≤ *n* ∧ *length l* ≥ *2* ⟶ (*multi-plus l ω* ⟷ (∃ *r*. *Some ω* = (*List.hd l*) ⊕ *r* ∧ *multi-plus* (*List.tl l*) *r*))
⟨*proof*⟩

lemma *not-multi-plus-empty*:
¬ *multi-plus* [] *ω*
⟨*proof*⟩

lemma *multi-plus-deter*:
length l ≤ *n* ⟹ *multi-plus l ω* ⟹ *multi-plus l ω'* ⟹ *ω* = *ω'*

⟨*proof*⟩

lemma *multi-plus-implies-multi-plus-of-drop*:
assumes *multi-plus l ω*
and *n* < *length l*
shows ∃ *a*. *multi-plus* (*drop n l*) *a* ∧ *ω* ≽ *a*
⟨*proof*⟩

lemma *multi-plus-bigger-than-head*:
assumes *length l* > *0*
and *multi-plus l ω*
shows *ω* ≽ *List.hd l*
⟨*proof*⟩

lemma *multi-plus-bigger*:
assumes *i* < *length l*
and *multi-plus l ω*
shows *ω* ≽ (*l* ! *i*)
⟨*proof*⟩

lemma *sum-then-singleton*:
Some a = *b* ⊕ *c* ⟷ {*a*} = {*b*} ⊗ {*c*} (**is** *?A* ⟷ *?B*)
⟨*proof*⟩

lemma *empty-set-sum*:
{} ⊗ *A* = {}
⟨*proof*⟩

end

end

2 Package Logic

In this section, we define our package logic, as described in [2], and then prove that this logic is sound and complete for packaging magic wands.

theory *PackageLogic*
imports *Main SepAlgebra*
begin

2.1 Definitions

type-synonym *'a abool* = *'a* ⇒ *bool*

datatype *'a aassertion* =
AStar 'a aassertion 'a aassertion

| *AImp* *'a abool 'a aassertion*
| *ASem* *'a abool*

locale *package-logic* = *sep-algebra* +

fixes *unit* :: *'a*
fixes *stable* :: *'a* ⇒ *bool*

assumes *unit-neutral*: *Some a* = *a* ⊕ *unit*
and *stable-sum*: *stable a* ⟹ *stable b* ⟹ *Some x* = *a* ⊕ *b* ⟹ *stable x*
and *stable-unit*: *stable unit*

begin

fun *sat* :: *'a aassertion* ⇒ *'a* ⇒ *bool* **where**
sat (*AStar A B*) *φ* ⟷ (∃ *a b*. *Some φ* = *a* ⊕ *b* ∧ *sat A a* ∧ *sat B b*)
| *sat* (*AImp b A*) *φ* ⟷ (*b φ* ⟶ *sat A φ*)
| *sat* (*ASem A*) *φ* ⟷ *A φ*

definition *mono-pure-cond* **where**
mono-pure-cond b ⟷ (∀ *φ*. *b φ* ⟷ *b* |*φ*|) ∧ (∀ *φ' φ r*. *pure r* ∧ *Some φ'* = *φ* ⊕ *r* ∧ ¬ *b φ* ⟶ ¬ *b φ'*)

definition *bool-conj* **where**
bool-conj a b x ⟷ *a x* ∧ *b x*

type-synonym *'c pruner* = *'c* ⇒ *bool*

definition *mono-pruner* :: *'a pruner* ⇒ *bool* **where**
mono-pruner p ⟷ (∀ *φ' φ r*. *pure r* ∧ *p φ* ∧ *Some φ'* = *φ* ⊕ *r* ⟶ *p φ'*)

fun *wf-assertion* **where**
wf-assertion (*AStar A B*) ⟷ *wf-assertion A* ∧ *wf-assertion B*
| *wf-assertion* (*AImp b A*) ⟷ *mono-pure-cond b* ∧ *wf-assertion A*
| *wf-assertion* (*ASem A*) ⟷ *mono-pruner A*

type-synonym *'c transformer* = *'c* ⇒ *'c*

type-synonym *'c ext-state* = *'c* × *'c* × *'c transformer*

inductive *package-rhs* ::
'a ⇒ *'a* ⇒ *'a ext-state set* ⇒ *'a abool* ⇒ *'a aassertion* ⇒ *'a* ⇒ *'a* ⇒ *'a ext-state set* ⇒ *bool* **where**

AStar: ⟦ *package-rhs φ f S pc A φ' f' S'* ; *package-rhs φ' f' S' pc B φ'' f'' S''* ⟧ ⟹ *package-rhs φ f S pc* (*AStar A B*) *φ'' f'' S''*
| *AImp*: *package-rhs φ f S* (*bool-conj pc b*) *A φ' f' S'* ⟹ *package-rhs φ f S pc*

$(AImp\ b\ A)\ \varphi'\ f'\ S'$

| *ASem*: $[\![\bigwedge a\ u\ T\ b.\ (a,\ u,\ T) \in S \Longrightarrow pc\ a \Longrightarrow b = witness\ (a,\ u,\ T) \Longrightarrow a \succeq b \wedge B\ b$;
$S' = \{\ (a,\ u,\ T)\ |a\ u\ T.\ (a,\ u,\ T) \in S \wedge \neg\ pc\ a\ \}$
$\cup\ \{\ (a \ominus b,\ the\ (u \oplus b),\ T)\ |a\ u\ T\ b.\ (a,\ u,\ T) \in S \wedge pc\ a \wedge b = witness\ (a,\ u,\ T)\ \}\ ]\!]$
$\Longrightarrow$ *package-rhs* $\varphi\ f\ S\ pc\ (ASem\ B)\ \varphi\ f\ S'$

| *AddFromOutside*: $[\![$ *Some* $\varphi = \varphi' \oplus m$; *package-rhs* $\varphi'\ f'\ S'\ pc\ A\ \varphi''\ f''\ S''$; *stable* m ; *Some* $f' = f \oplus m$;
$S' = \{\ (r,\ u,\ T)\ |a\ u\ T\ r.\ (a,\ u,\ T) \in S \wedge Some\ r = a \oplus (T\ f' \ominus T\ f)\ \wedge r\ \#\#\ u\ \}\ ]\!]$
$\Longrightarrow$ *package-rhs* $\varphi\ f\ S\ pc\ A\ \varphi''\ f''\ S''$

definition *package-sat* **where**
package-sat pc A a' u' u $\longleftrightarrow$ $(pc\ |a'| \longrightarrow (\exists x.\ Some\ x = |a'| \oplus (u' \ominus u\ ) \wedge sat\ A\ x))$

definition *package-rhs-connection* :: *'a* $\Rightarrow$ *'a* $\Rightarrow$ *'a ext-state set* $\Rightarrow$ *'a abool* $\Rightarrow$ *'a aassertion* $\Rightarrow$ *'a* $\Rightarrow$ *'a* $\Rightarrow$ *'a ext-state set* $\Rightarrow$ *bool* **where**
package-rhs-connection $\varphi\ f\ S\ pc\ A\ \varphi'\ f'\ S' \longleftrightarrow f' \succeq f \wedge \varphi\ \#\#\ f \wedge \varphi \oplus f = \varphi' \oplus f' \wedge stable\ f' \wedge$
$(\forall (a,\ u,\ T) \in S.\ \exists au.\ Some\ au = a \oplus u \wedge (au\ \#\#\ (T\ f' \ominus T\ f) \longrightarrow$
$(\exists\, a'\ u'.\ (a',\ u',\ T) \in S' \wedge |a'| \succeq |a| \wedge au \oplus (T\ f' \ominus T\ f) = a' \oplus u' \wedge u' \succeq u \wedge package\text{-}sat\ pc\ A\ a'\ u'\ u)))$

definition *mono-transformer* :: *'a transformer* $\Rightarrow$ *bool* **where**
mono-transformer $T \longleftrightarrow (\forall\, \varphi\ \varphi'.\ \varphi' \succeq \varphi \longrightarrow T\ \varphi' \succeq T\ \varphi) \wedge T\ unit = unit$

definition *valid-package-set* **where**
valid-package-set $S\ f \longleftrightarrow (\forall\, (a,\ u,\ T) \in S.\ a\ \#\#\ u \wedge |a| \succeq |u| \wedge mono\text{-}transformer\ T \wedge a \succeq |T\ f|\ )$

definition *intuitionistic* **where**
intuitionistic $A \longleftrightarrow (\forall\, \varphi'\ \varphi.\ \varphi' \succeq \varphi \wedge A\ \varphi \longrightarrow A\ \varphi')$

definition *pure-remains* **where**
pure-remains $S \longleftrightarrow (\forall\, (a,\ u,\ p) \in S.\ pure\ a)$

definition *is-footprint-general* :: *'a* $\Rightarrow$ (*'a* $\Rightarrow$ *'a* $\Rightarrow$ *'a*) $\Rightarrow$ *'a aassertion* $\Rightarrow$ *'a aassertion* $\Rightarrow$ *bool* **where**
is-footprint-general w R A B $\longleftrightarrow$ $(\forall\, a\ b.\ sat\ A\ a \wedge Some\ b = a \oplus R\ a\ w \longrightarrow sat\ B\ b)$

definition *is-footprint-standard* :: *'a* $\Rightarrow$ *'a aassertion* $\Rightarrow$ *'a aassertion* $\Rightarrow$ *bool* **where**
is-footprint-standard w A B $\longleftrightarrow$ $(\forall\, a\ b.\ sat\ A\ a \wedge Some\ b = a \oplus w \longrightarrow sat\ B\ b)$

2.2 Lemmas

lemma *is-footprint-generalI*:
assumes $\bigwedge a\ b.\ sat\ A\ a \wedge Some\ b = a \oplus R\ a\ w \Longrightarrow sat\ B\ b$
shows *is-footprint-general w R A B*
⟨*proof*⟩

lemma *is-footprint-standardI*:
assumes $\bigwedge a\ b.\ sat\ A\ a \wedge Some\ b = a \oplus w \Longrightarrow sat\ B\ b$
shows *is-footprint-standard w A B*
⟨*proof*⟩

lemma *mono-pure-condI*:
assumes $\bigwedge \varphi.\ b\ \varphi \longleftrightarrow b\ |\varphi|$
and $\bigwedge \varphi\ \varphi'\ r.\ pure\ r \wedge Some\ \varphi' = \varphi \oplus r \wedge \neg\ b\ \varphi \Longrightarrow \neg\ b\ \varphi'$
shows *mono-pure-cond b*
⟨*proof*⟩

lemma *mono-pure-cond-conj*:
assumes *mono-pure-cond pc*
and *mono-pure-cond b*
shows *mono-pure-cond* (*bool-conj pc b*)
⟨*proof*⟩

lemma *bigger-sum*:
assumes $Some\ \varphi = a \oplus b$
and $\varphi' \succeq \varphi$
shows $\exists\, b'.\ b' \succeq b \wedge Some\ \varphi' = a \oplus b'$
⟨*proof*⟩

lemma *wf-assertion-sat-larger-pure*:
assumes *wf-assertion A*
and *sat A* φ
and $Some\ \varphi' = \varphi \oplus r$
and *pure r*
shows *sat A* φ'
⟨*proof*⟩

lemma *package-satI*:
assumes $pc\ |a'| \Longrightarrow (\exists\, x.\ Some\ x = |a'| \oplus (u' \ominus u\ ) \wedge sat\ A\ x)$
shows *package-sat pc A a' u' u*
⟨*proof*⟩

lemma *package-rhs-connection-instantiate*:
assumes *package-rhs-connection* φ *f S pc A* φ' *f' S'*
and $(a,\ u,\ T) \in S$
obtains *au* **where** $Some\ au = a \oplus u$

and $au \mathbin{\#\#} (T\, f' \ominus T\, f) \Longrightarrow \exists\, a'\, u'.\ (a',\ u',\ T) \in S' \wedge |a'| \succeq |a| \wedge au \oplus (T\, f' \ominus T\, f) = a' \oplus u' \wedge u' \succeq u \wedge$ *package-sat pc A* $a'\ u'\ u$

⟨*proof*⟩

lemma *package-rhs-connectionI*:
assumes $\varphi \oplus f = \varphi' \oplus f'$
and *stable* f'
and $\varphi \mathbin{\#\#} f$
and $f' \succeq f$
and $\bigwedge a\ u\ T.\ (a,\ u,\ T) \in S \Longrightarrow (\exists\, au.\ \mathit{Some}\ au = a \oplus u \wedge (au \mathbin{\#\#} (T\, f' \ominus T\, f) \longrightarrow$
$(\exists\, a'\, u'.\ (a',\ u',\ T) \in S' \wedge |a'| \succeq |a| \wedge au \oplus (T\, f' \ominus T\, f) = a' \oplus u' \wedge u' \succeq u \wedge$ *package-sat pc A* $a'\ u'\ u)))$
shows *package-rhs-connection* $\varphi\ f\ S$ *pc A* $\varphi'\ f'\ S'$

⟨*proof*⟩

lemma *valid-package-setI*:
assumes $\bigwedge a\ u\ T.\ (a,\ u,\ T) \in S \Longrightarrow a \mathbin{\#\#} u \wedge |a| \succeq |u| \wedge$ *mono-transformer* $T \wedge a \succeq |T\, f|$
shows *valid-package-set* $S\ f$

⟨*proof*⟩

lemma *defined-sum-move*:
assumes $a \mathbin{\#\#} b$
and $\mathit{Some}\ b = x \oplus y$
and $\mathit{Some}\ a' = a \oplus x$
shows $a' \mathbin{\#\#} y$

⟨*proof*⟩

lemma *bigger-core-sum-defined*:
assumes $|a| \succeq b$
shows $\mathit{Some}\ a = a \oplus b$

⟨*proof*⟩

lemma *package-rhs-proof*:
assumes *package-rhs* $\varphi\ f\ S$ *pc A* $\varphi'\ f'\ S'$
and *valid-package-set* $S\ f$
and *wf-assertion A*
and *mono-pure-cond pc*
and *stable* f
and $\varphi \mathbin{\#\#} f$
shows *package-rhs-connection* $\varphi\ f\ S$ *pc A* $\varphi'\ f'\ S' \wedge$ *valid-package-set* $S'\ f'$

⟨*proof*⟩

lemma *unit-core*:
$|\mathit{unit}| = \mathit{unit}$

⟨*proof*⟩

lemma *unit-smaller*:
$\varphi \succeq$ *unit*
⟨*proof*⟩

2.3 Lemmas for completeness

lemma *sat-star-exists-bigger*:
assumes *sat* (*AStar A B*) φ
and *wf-assertion A*
and *wf-assertion B*
shows $\exists\, a\ b.\ |a| \succeq |\varphi| \wedge |b| \succeq |\varphi| \wedge$ *Some* $\varphi = a \oplus b \wedge$ *sat A a* $\wedge$ *sat B b*
⟨*proof*⟩

lemma *let-pair-instantiate*:
assumes $(a,\ b) = f\ x\ y$
shows (*let* $(a,\ b) = f\ x\ y$ *in* $g\ a\ b) = g\ a\ b$
⟨*proof*⟩

lemma *greater-than-sum-exists*:
assumes $a \succeq b$
and *Some* $b = b1 \oplus b2$
shows $\exists\, r.$ *Some* $a = r \oplus b2 \wedge |r| \succeq |a| \wedge r \succeq b1$
⟨*proof*⟩

lemma *bigger-the*:
assumes *Some* $a = x' \oplus y$
and $x' \succeq x$
shows *the* ($|a|$ $\oplus$ x') $\succeq$ *the* ($|a|$ $\oplus$ x)
⟨*proof*⟩

lemma *wf-assertion-and-the*:
assumes $|a|$ ## b
and *sat A b*
and *wf-assertion A*
shows *sat A* (*the* ($|a| \oplus b$))
⟨*proof*⟩

lemma *minus-some*:
assumes $a \succeq b$
shows *Some* $a = b \oplus (a \ominus b)$
⟨*proof*⟩

lemma *core-mono*:
assumes $a \succeq b$
shows $|a| \succeq |b|$
⟨*proof*⟩

lemma *prove-last-completeness*:

assumes $a' \succeq a$
and *Some* $a = nf1 \oplus f2$
shows $a' \ominus nf1 \succeq f2$
⟨*proof*⟩

lemma *completeness-aux*:
assumes $\bigwedge a\ u\ T.\ (a,\ u,\ T) \in S \Longrightarrow |f1\ a\ u\ T| \succeq |a| \wedge$ *Some* $a = f1\ a\ u\ T \oplus f2\ a\ u\ T \wedge (pc\ |a| \longrightarrow sat\ A\ (the\ (\ |a| \oplus (f1\ a\ u\ T))))$
and *valid-package-set* $S\ f$
and *wf-assertion* A
and *mono-pure-cond* pc
and *stable* $f \wedge \varphi\ \#\#\ f$
shows $\exists S'.$ *package-rhs* $\varphi\ f\ S\ pc\ A\ \varphi\ f\ S' \wedge (\forall (a',\ u',\ T) \in S'.\ \exists a\ u.\ (a,\ u,\ T) \in S \wedge a' \succeq f2\ a\ u\ T \wedge |a'| = |a|\ )$
⟨*proof*⟩

2.4 Soundness

theorem *general-soundness*:
assumes *package-rhs* φ *unit* $\{\ (a,\ unit,\ T)\ |a\ T.\ (a,\ T) \in S\ \}$ (λ-. *True*) $A\ \varphi'\ f\ S'$
and $\bigwedge a\ T.\ (a,\ T) \in S \Longrightarrow$ *mono-transformer* T
and *wf-assertion* A
and *intuitionistic* $(sat\ A) \vee$ *pure-remains* S'
shows *Some* $\varphi = \varphi' \oplus f \wedge$ *stable* $f \wedge (\forall (a,\ T) \in S.\ a\ \#\#\ T\ f \longrightarrow sat\ A\ (the\ (a \oplus T\ f)))$
⟨*proof*⟩

theorem *soundness*:
assumes *wf-assertion* B
and $\bigwedge a.$ *sat* $A\ a \Longrightarrow a \in S$
and $\bigwedge a.\ a \in S \Longrightarrow$ *mono-transformer* $(R\ a)$
and *package-rhs* σ *unit* $\{\ (a,\ unit,\ R\ a)\ |a.\ a \in S\ \}$ (λ-. *True*) $B\ \sigma'\ w\ S'$
and *intuitionistic* $(sat\ B) \vee$ *pure-remains* S'
shows *stable* $w \wedge$ *Some* $\sigma = \sigma' \oplus w \wedge$ *is-footprint-general* $w\ R\ A\ B$
⟨*proof*⟩

corollary *soundness-paper*:
assumes *wf-assertion* B
and $\bigwedge a.$ *sat* $A\ a \Longrightarrow a \in S$
and *package-rhs* σ *unit* $\{\ (a,\ unit,\ id)\ |a.\ a \in S\ \}$ (λ-. *True*) $B\ \sigma'\ w\ S'$
and *intuitionistic* $(sat\ B) \vee$ *pure-remains* S'
shows *stable* $w \wedge$ *Some* $\sigma = \sigma' \oplus w \wedge$ *is-footprint-standard* $w\ A\ B$
⟨*proof*⟩

2.5 Completeness

theorem *general-completeness*:
assumes $\bigwedge a\ u\ T\ x.\ (a,\ u,\ T) \in S \Longrightarrow$ *Some* $x = a \oplus T\ f \Longrightarrow sat\ A\ x$
and *Some* $\varphi = \varphi' \oplus f$

and *stable f*
and *valid-package-set S unit*
and *wf-assertion A*
shows $\exists S'$. *package-rhs* φ *unit S* (λ-. *True*) *A* φ' *f S'*
⟨*proof*⟩

theorem *completeness*:
assumes *wf-assertion B*
and *stable w* $\wedge$ *is-footprint-general w R A B*
and *Some* $\sigma = \sigma' \oplus w$
and $\bigwedge a$. *sat A a* $\Longrightarrow$ *mono-transformer* (*R a*)
shows $\exists S'$. *package-rhs* σ *unit* $\{(a, unit, R\ a)\ |a.\ sat\ A\ a\}$ (λ-. *True*) *B* σ' *w S'*
⟨*proof*⟩

corollary *completeness-paper*:
assumes *wf-assertion B*
and *stable w* $\wedge$ *is-footprint-standard w A B*
and *Some* $\sigma = \sigma' \oplus w$
shows $\exists S'$. *package-rhs* σ *unit* $\{(a, unit, id)\ |a.\ sat\ A\ a\}$ (λ-. *True*) *B* σ' *w S'*
⟨*proof*⟩

end

end

References

[1] C. Calcagno, P. W. O'Hearn, and H. Yang. Local Action and Abstract Separation Logic. In *Logic in Computer Science (LICS)*, pages 366–375, 2007.

[2] T. Dardinier, G. Parthasarathy, N. Weeks, P. Müller, and A. J. Summers. Sound Automation of Magic Wands. In *Computer Aided Verification (CAV)*, 2022. To appear.

[3] R. Dockins, A. Hobor, and A. W. Appel. A Fresh Look at Separation Algebras and Share Accounting. In Z. Hu, editor, *Asian Symposium on Programming Languages and Systems (APLAS)*, pages 161–177, 2009.

[4] J. C. Reynolds. Separation Logic: A Logic for Shared Mutable Data Structures. In *Logic in Computer Science (LICS)*, pages 55–74. IEEE, 2002.