Unifying heterogeneous state-spaces with lenses

Simon Foster[1], Frank Zeyda[2], and Jim Woodcock[1]

[1] University of York
[2] Teesside University

ICTAC 2016

25th of October 2016

Table of Contents

Table of Contents

UTP in brief

- framework for formulation of denotational semantic models
- based on the idea of programs-as-predicates
- predicates encode the set of observable behaviours
- alphabetised relational calculus – models expressed as relations
- alpha predicates $(\alpha P, P)$ over input, output variables (x / x')
- alphabet gives the domain of possible observations

UTP in brief

- framework for formulation of denotational semantic models
- based on the idea of programs-as-predicates
- predicates encode the set of observable behaviours
- alphabetised relational calculus – models expressed as relations
- alpha predicates $(\alpha P, P)$ over input, output variables (x / x')
- alphabet gives the domain of possible observations

$$
\begin{aligned}
x := v &\triangleq x' = v \wedge y' = y \\
P \,;\, Q &\triangleq \exists\, x_0 \bullet P[x_0/x'] \wedge Q[x_0/x] \\
P \lhd b \rhd Q &\triangleq (b \wedge P) \vee (\neg b \wedge Q) \\
P^{*} &\triangleq \nu X \bullet P \,;\, X
\end{aligned}
$$

UTP in brief

- framework for formulation of denotational semantic models
- based on the idea of programs-as-predicates
- predicates encode the set of observable behaviours
- alphabetised relational calculus – models expressed as relations
- alpha predicates $(\alpha P, P)$ over input, output variables (x / x')
- alphabet gives the domain of possible observations

$$
\begin{aligned}
x := v &\triangleq x' = v \wedge y' = y \\
P \,;\, Q &\triangleq \exists\, x_0 \bullet P[x_0/x'] \wedge Q[x_0/x] \\
P \lhd b \rhd Q &\triangleq (b \wedge P) \vee (\neg b \wedge Q) \\
P^{*} &\triangleq \nu X \bullet P \,;\, X
\end{aligned}
$$

- avoids fixing a language's abstract syntax tree
- enables composition of semantic models (via UTP theories)

Algebraic laws of programs

$$
\begin{aligned}
(P \,;\, Q) \,;\, R &= P \,;\, (Q \,;\, R) \\
P \,;\, \mathbf{false} &= \mathbf{false} \,;\, P = \mathbf{false} \\
(P \lhd b \rhd Q) \,;\, R &= (P \,;\, R) \lhd b \rhd (Q \,;\, R) \\
\mathbf{while}\ b\ \mathbf{do}\ P &= (P \,;\, \mathbf{while}\ b\ \mathbf{do}\ P) \lhd b \rhd I\!I \\
P \,;\, Q &= \exists\, x_0 \bullet P[x/x_0] \,;\, P[x'/x_0] \\
(P \wedge b) \,;\, Q &= P \,;\, (b' \wedge Q) \\
I\!I_{\{x,x'\} \cup A} &= (x = x') \wedge I\!I_A \\
(x := e \,;\, y := f) &= (y := f \,;\, x := e)^{(1)} \\
x := e \,;\, P &= P[e/x]
\end{aligned}
$$

(1) $x \neq y, x \notin fv(f), y \notin fv(e)$

Algebraic laws of programs

$$
\begin{aligned}
(P \,;\, Q) \,;\, R &= P \,;\, (Q \,;\, R) \\
P \,;\, \mathbf{false} &= \mathbf{false} \,;\, P = \mathbf{false} \\
(P \lhd b \rhd Q) \,;\, R &= (P \,;\, R) \lhd b \rhd (Q \,;\, R) \\
\mathbf{while}\, b \,\mathbf{do}\, P &= (P \,;\, \mathbf{while}\, b \,\mathbf{do}\, P) \lhd b \rhd \mathit{II} \\
P \,;\, Q &= \exists\, x_0 \bullet P[x/x_0] \,;\, P[x'/x_0] \\
(P \wedge b) \,;\, Q &= P \,;\, (b' \wedge Q) \\
\mathit{II}_{\{x,x'\} \cup A} &= (x = x') \wedge \mathit{II}_A \\
(x := e \,;\, y := f) &= (y := f \,;\, x := e)^{(1)} \\
x := e \,;\, P &= P[e/x]
\end{aligned}
$$

(1) $x \neq y, x \notin fv(f), y \notin fv(e)$

Algebraic laws of programs

$$
\begin{aligned}
(P \,;\, Q) \,;\, R &= P \,;\, (Q \,;\, R) \\
P \,;\, \mathbf{false} &= \mathbf{false} \,;\, P \;=\; \mathbf{false} \\
(P \lhd b \rhd Q) \,;\, R &= (P \,;\, R) \lhd b \rhd (Q \,;\, R) \\
\mathbf{while}\ b\ \mathbf{do}\ P &= (P \,;\, \mathbf{while}\ b\ \mathbf{do}\ P) \lhd b \rhd I\!I \\
P \,;\, Q &= \exists\, x_0 \bullet P[x/x_0] \,;\, P[x'/x_0] \\
(P \wedge b) \,;\, Q &= P \,;\, (b' \wedge Q) \\
I\!I_{\{x,x'\} \cup A} &= (x = x') \wedge I\!I_A \\
(x := e \,;\, y := f) &= (y := f \,;\, x := e)^{(1)} \\
x := e \,;\, P &= P[e/x]
\end{aligned}
$$

(1) $x \neq y, x \notin fv(f), y \notin fv(e)$

Algebraic laws of programs

$$
\begin{aligned}
(P \,;\, Q) \,;\, R &= P \,;\, (Q \,;\, R) \\
P \,;\, \textbf{false} &= \textbf{false} \,;\, P \;=\; \textbf{false} \\
(P \lhd b \rhd Q) \,;\, R &= (P \,;\, R) \lhd b \rhd (Q \,;\, R) \\
\textbf{while}\; b \;\textbf{do}\; P &= (P \,;\, \textbf{while}\; b \;\textbf{do}\; P) \lhd b \rhd I\!I \\
P \,;\, Q &= \exists\, x_0 \bullet P[x/x_0] \,;\, P[x'/x_0] \\
(P \wedge b) \,;\, Q &= P \,;\, (b' \wedge Q) \\
I\!I_{\{x,x'\} \cup A} &= (x = x') \wedge I\!I_A \\
(x := e \,;\, y := f) &= (y := f \,;\, x := e)^{(1)} \\
x := e \,;\, P &= P[e/x]
\end{aligned}
$$

(1) $x \neq y, x \notin fv(f), y \notin fv(e)$

Predicate model

- predicates modelled as sets of observations: $\mathbb{P}\,\mathcal{S}$

Predicate model

- predicates modelled as sets of observations: $\mathbb{P}\,\mathcal{S}$
- the observations that satisfy the predicate
- how to model the state space $\mathcal{S}$ of these predicates?

Predicate model

- predicates modelled as sets of observations: $\mathbb{P}\,\mathcal{S}$
- the observations that satisfy the predicate
- how to model the state space $\mathcal{S}$ of these predicates?
- precise mathematical model required for mechanisation etc.
- e.g. bindings ($\mathcal{B}$) describe states as variable valuations

$$\mathcal{B} = \{f : \mathit{Var} \to \mathit{Val} \mid \forall\, x \in \mathrm{dom}(f) \bullet f(x) : x_\tau\}$$

Predicate model

- predicates modelled as sets of observations: $\mathbb{P}\,\mathcal{S}$
- the observations that satisfy the predicate
- how to model the state space $\mathcal{S}$ of these predicates?
- precise mathematical model required for mechanisation etc.
- e.g. bindings ($\mathcal{B}$) describe states as variable valuations

$$\mathcal{B} = \{f : \mathit{Var} \rightarrow \mathit{Val} \mid \forall\, x \in \mathrm{dom}(f) \bullet f(x) : x_\tau\}$$

- but what are the meanings of Var and Val?
- need to fix the syntax of variables and values upfront
- what about naming problems, e.g. α-conversion and aliasing?
- can we reason about $\mathcal{S}$ more generically?

Modelling state with lenses

- lenses as a uniform semantic interface for variables

Modelling state with lenses

- lenses as a uniform semantic interface for variables

- represent $\mathcal{S}$ as suitable Isabelle types (e.g. records)
- identify variables by the position they occupy in the state
- regions of the state can be variously related
- nameless and spatial representation of variables

Modelling state with lenses

- lenses as a uniform semantic interface for variables

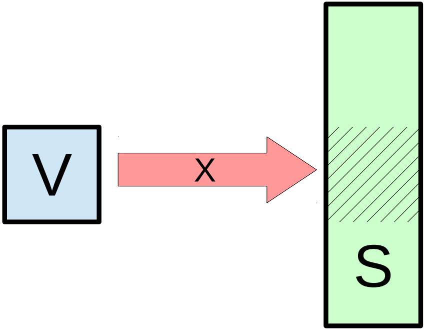
- represent $\mathcal{S}$ as suitable Isabelle types (e.g. records)
- identify variables by the position they occupy in the state
- regions of the state can be variously related
- nameless and spatial representation of variables

- operators for transforming and decomposing a state space
- enable algebraic account of state
- approx. meta-logic operators (fresh variables, substitution)
- from this basis can prove UTP's fundamental laws
- theory of lenses requires only first-order polymorphic typing

Table of Contents

What is a lens?

- $X : V \Longrightarrow S$ for view type V and (“bigger”) source type S
- allow to focus on V independently of rest of S



What is a lens?

- $X : V \Longrightarrow S$ for view type V and ("bigger") source type S
- allow to focus on V independently of rest of S
- signature consists of two functions:
  - $\mathit{get} : S \to V$
  - $\mathit{put} : S \to V \to S$
- characterised through a number of algebraic laws

What is a lens?

- $X : V \Longrightarrow S$ for view type V and ("bigger") source type S
- allow to focus on V independently of rest of S
- signature consists of two functions:
  - $\mathit{get} : S \to V$
  - $\mathit{put} : S \to V \to S$
- characterised through a number of algebraic laws
- originally created to characterise bidirectional transformations
- example: record lenses

$$\llparenthesis\ \mathit{forename} : \mathit{String}, \mathit{surname} : \mathit{String}, \mathit{age} : \mathit{Int}\ \rrparenthesis$$

What is a lens?

- $X : V \Longrightarrow S$ for view type V and ("bigger") source type S
- allow to focus on V independently of rest of S
- signature consists of two functions:
  - $\mathit{get} : S \to V$
  - $\mathit{put} : S \to V \to S$
- characterised through a number of algebraic laws
- originally created to characterise bidirectional transformations
- example: record lenses

$$\llparenthesis\, \underbrace{\mathit{forename} : \mathit{String}}_{\text{lens 1}}, \underbrace{\mathit{surname} : \mathit{String}}_{\text{lens 2}}, \underbrace{\mathit{age} : \mathit{Int}}_{\text{lens 3}} \,\rrparenthesis$$

What is a lens?

- $X : V \Longrightarrow S$ for view type V and ("bigger") source type S
- allow to focus on V independently of rest of S
- signature consists of two functions:
  - $\mathit{get} : S \to V$
  - $\mathit{put} : S \to V \to S$
- characterised through a number of algebraic laws
- originally created to characterise bidirectional transformations
- example: record lenses

$$\llparenthesis\, \underbrace{\mathit{forename} : \mathit{String}, \mathit{surname} : \mathit{String}}_{\text{lens 4}}, \mathit{age} : \mathit{Int} \,\rrparenthesis$$

Lens laws

$$\mathit{get}\,(\mathit{put}\,s\,v) = v \qquad \text{(PutGet)}$$

$$\mathit{put}\,(\mathit{put}\,s\,v')\,v = \mathit{put}\,s\,v \qquad \text{(PutPut)}$$

$$\mathit{put}\,s\,(\mathit{get}\,s) = s \qquad \text{(GetPut)}$$

Lens laws

$$\mathit{get}\,(\mathit{put}\,s\,v) = v \qquad \text{(PutGet)}$$

$$\mathit{put}\,(\mathit{put}\,s\,v')\,v = \mathit{put}\,s\,v \qquad \text{(PutPut)}$$

$$\mathit{put}\,s\,(\mathit{get}\,s) = s \qquad \text{(GetPut)}$$

- PutGet + GetPut characterise well-behaved lenses
- addition of PutPut characterise very well-behaved lenses

Lens laws

$$\mathit{get}\,(\mathit{put}\,s\,v) = v \qquad \text{(PutGet)}$$

$$\mathit{put}\,(\mathit{put}\,s\,v')\,v = \mathit{put}\,s\,v \qquad \text{(PutPut)}$$

$$\mathit{put}\,s\,(\mathit{get}\,s) = s \qquad \text{(GetPut)}$$

- PutGet + GetPut characterise well-behaved lenses
- addition of PutPut characterise very well-behaved lenses

$$\mathit{put}\,s\,(\mathit{get}\,s') = s' \qquad \text{(StrongGetPut)}$$

Lens laws

$$\mathit{get}\,(\mathit{put}\,s\,v) = v \qquad \text{(PutGet)}$$

$$\mathit{put}\,(\mathit{put}\,s\,v')\,v = \mathit{put}\,s\,v \qquad \text{(PutPut)}$$

$$\mathit{put}\,s\,(\mathit{get}\,s) = s \qquad \text{(GetPut)}$$

- PutGet + GetPut characterise well-behaved lenses
- addition of PutPut characterise very well-behaved lenses

$$\mathit{put}\,s\,(\mathit{get}\,s') = s' \qquad \text{(StrongGetPut)}$$

- StrongGetPut characterises bijective lenses

Lens laws

$$\mathit{get}\,(\mathit{put}\,s\,v) = v \qquad \text{(PutGet)}$$

$$\mathit{put}\,(\mathit{put}\,s\,v')\,v = \mathit{put}\,s\,v \qquad \text{(PutPut)}$$

$$\mathit{put}\,s\,(\mathit{get}\,s) = s \qquad \text{(GetPut)}$$

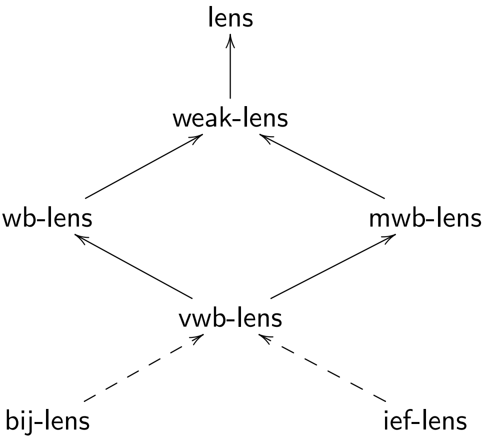
- PutGet + GetPut characterise well-behaved lenses
- addition of PutPut characterise very well-behaved lenses

$$\mathit{put}\,s\,(\mathit{get}\,s') = s' \qquad \text{(StrongGetPut)}$$

- StrongGetPut characterises bijective lenses
- we add
  - weak lenses (PutGet)
  - mainly well-behaved lenses (PutGet + PutPut)
  - ineffectual lenses

Algebraic hierarchy

- formalised using locales in Isabelle



Models

- records – each field
- total functions – each domain element
- partial functions (mwb-lens)
- lists (mwb-lens) – each index

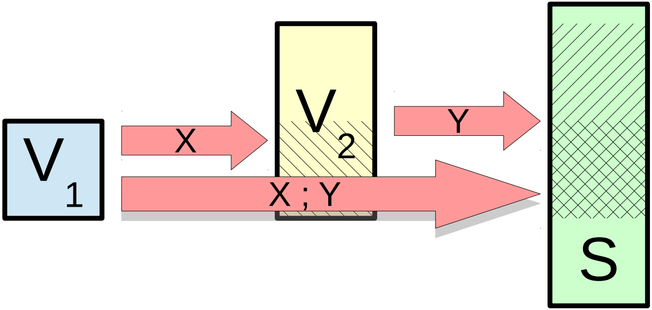
Models

- records – each field
- total functions – each domain element
- partial functions (mwb-lens)
- lists (mwb-lens) – each index
- each are potential state models
- e.g. partial functions to model heaps

Table of Contents

Lens composition

- $X \fatsemi Y$: identify the source of X with the view of Y
- e.g. view a record field within a field



Lens composition

- $X \fatsemi Y$: identify the source of X with the view of Y
- e.g. view a record field within a field
- definition simply composes the *get* and *put* functions

Lens composition

- $X \mathbin{;} Y$: identify the source of X with the view of Y
- e.g. view a record field within a field
- definition simply composes the *get* and *put* functions
- $\mathbin{;}$ is closed under weak-lens, wb-lens, mwb-lens, and vwb-lens

Lens composition

- $X \fatsemi Y$: identify the source of X with the view of Y
- e.g. view a record field within a field
- definition simply composes the *get* and *put* functions
- $\fatsemi$ is closed under weak-lens, wb-lens, mwb-lens, and vwb-lens
- $\mathbf{1} : S \Longrightarrow S$: the (bijective) identity lens

Lens composition

- $X \fatsemi Y$: identify the source of X with the view of Y
- e.g. view a record field within a field
- definition simply composes the *get* and *put* functions
- $\fatsemi$ is closed under weak-lens, wb-lens, mwb-lens, and vwb-lens
- $\mathbf{1} : S \Longrightarrow S$: the (bijective) identity lens

$$(X \fatsemi Y) \fatsemi Z = X \fatsemi (Y \fatsemi Z)$$

$$\mathbf{1} \fatsemi X = X$$

$$X \fatsemi \mathbf{1} = X$$

Lens composition

- $X \fatsemi Y$: identify the source of X with the view of Y
- e.g. view a record field within a field
- definition simply composes the *get* and *put* functions
- $\fatsemi$ is closed under weak-lens, wb-lens, mwb-lens, and vwb-lens
- $\mathbf{1} : S \Longrightarrow S$: the (bijective) identity lens

$$(X \fatsemi Y) \fatsemi Z = X \fatsemi (Y \fatsemi Z)$$

$$\mathbf{1} \fatsemi X = X$$

$$X \fatsemi \mathbf{1} = X$$

- $\mathbf{0} : () \Longrightarrow S$: the (ineffectual) unit lens

Lens difference

- how to compare the behaviour of two or more lenses?

Lens difference

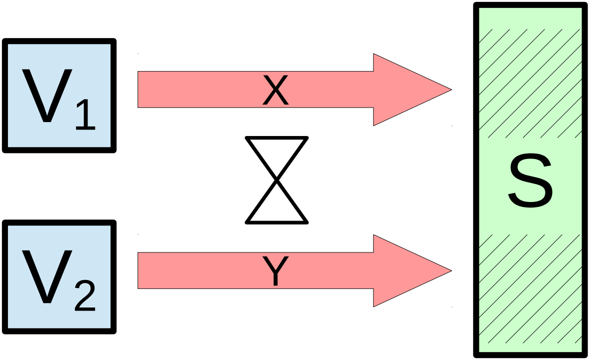
- how to compare the behaviour of two or more lenses?
- e.g. are two variables (behaviourally) identical?

Lens difference

- how to compare the behaviour of two or more lenses?
- e.g. are two variables (behaviourally) identical?

- lens independence ($X \bowtie Y$)
- X, Y are independent if they view spatially separate regions
- can give this a purely algebraic characterisation
- avoids syntactic aliasing issues

Lens independence visualised



Lens independence definition

- lenses X and Y are independent ($X \bowtie Y$) provided

Lens independence definition

- lenses X and Y are independent ($X \bowtie Y$) provided

$$\begin{aligned} \mathit{put}_X (\mathit{put}_Y\ s\ v)\ u &= \mathit{put}_Y (\mathit{put}_X\ s\ u)\ v \\ \mathit{get}_X (\mathit{put}_Y\ s\ v) &= \mathit{get}_X\, s \\ \mathit{get}_Y (\mathit{put}_X\ s\ u) &= \mathit{get}_Y\, s \end{aligned}$$

- for $s : S$, $u : V_1$, and $v : V_2$

Lens independence definition

- lenses X and Y are independent ($X \bowtie Y$) provided

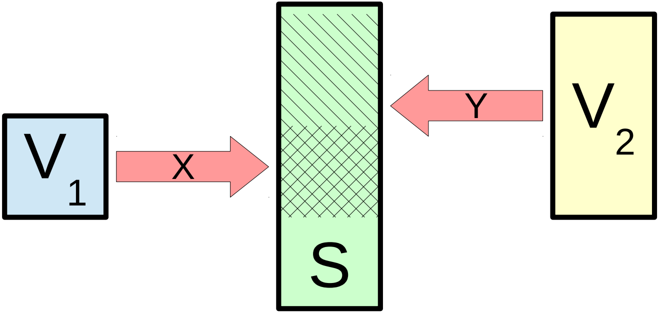
$$
\begin{aligned}
\mathit{put}_X(\mathit{put}_Y\ s\ v)\ u &= \mathit{put}_Y(\mathit{put}_X\ s\ u)\ v \\
\mathit{get}_X(\mathit{put}_Y\ s\ v) &= \mathit{get}_X\, s \\
\mathit{get}_Y(\mathit{put}_X\ s\ u) &= \mathit{get}_Y\, s
\end{aligned}
$$

- for $s : S$, $u : V_1$, and $v : V_2$
- $\bowtie$ is symmetric, and irreflexive for effectual lenses

$$
X \bowtie Y \Leftrightarrow Y \bowtie X
$$
$$
\mathbf{0} \bowtie X
$$

Sublens relation

- X is a sublens of Y ($X \preceq Y$) if Y's view encompasses X's



Sublens relation

- X is a sublens of Y ($X \preceq Y$) if Y's view encompasses X's
- there exists a "shim" lens Z which allows X to behave like Y

$$X \preceq Y \triangleq \exists\, Z.\, Z \in \textsf{wb-lens} \wedge X = Z \mathbin{;} Y$$

Sublens relation

- X is a sublens of Y ($X \preceq Y$) if Y's view encompasses X's
- there exists a "shim" lens Z which allows X to behave like Y

$$X \preceq Y \triangleq \exists\, Z.\, Z \in \textsf{wb-lens} \wedge X = Z \mathbin{;} Y$$

$$\begin{tikzcd}
 & S & \\
V_1 \arrow[ur, "X"] \arrow[rr, dashed, "Z"] & & V_2 \arrow[ul, "Y"']
\end{tikzcd}$$

Sublens relation

- X is a sublens of Y ($X \preceq Y$) if Y's view encompasses X's
- there exists a "shim" lens Z which allows X to behave like Y

$$X \preceq Y \triangleq \exists\, Z.\, Z \in \textsf{wb-lens} \land X = Z \fatsemi Y$$

- $\preceq$ is a preorder (reflexive, transitive)
- $\mathbf{0}$ is the least element, and $\mathbf{1}$ is the greatest element
- $\preceq$ thus orders the "effect" of a lens on a source

Sublens relation

- X is a sublens of Y ($X \preceq Y$) if Y's view encompasses X's
- there exists a "shim" lens Z which allows X to behave like Y

$$X \preceq Y \triangleq \exists Z.\, Z \in \mathsf{wb\text{-}lens} \wedge X = Z \fatsemi Y$$

- $\preceq$ is a preorder (reflexive, transitive)
- $\mathbf{0}$ is the least element, and $\mathbf{1}$ is the greatest element
- $\preceq$ thus orders the "effect" of a lens on a source
- can induce an equivalence rel on lenses: $\approx \triangleq \preceq \cap \supseteq_L$

Sublens relation

- X is a sublens of Y ($X \preceq Y$) if Y's view encompasses X's
- there exists a "shim" lens Z which allows X to behave like Y

$$X \preceq Y \triangleq \exists\, Z.\, Z \in \text{wb-lens} \wedge X = Z \fatsemi Y$$

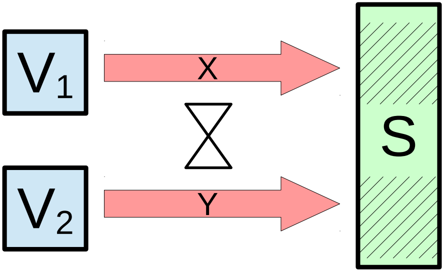
- $\preceq$ is a preorder (reflexive, transitive)
- $\mathbf{0}$ is the least element, and $\mathbf{1}$ is the greatest element
- $\preceq$ thus orders the "effect" of a lens on a source
- can induce an equivalence rel on lenses: $\approx \triangleq \preceq \cap \supseteq_L$

Theorem (Sublens preserves independence)

If $X \preceq Y$ *and* $Y \bowtie Z$ *then also* $X \bowtie Z$

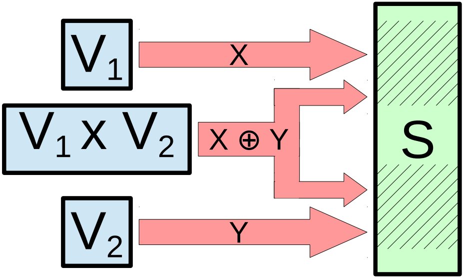
Lens sum

- $X \oplus Y$ parallel composes two independent lenses



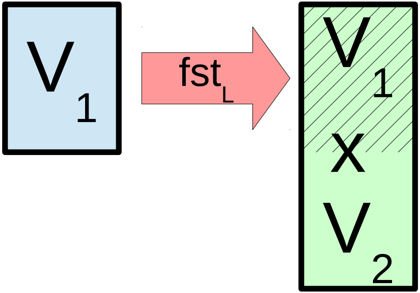
Lens sum

- $X \oplus Y$ parallel composes two independent lenses



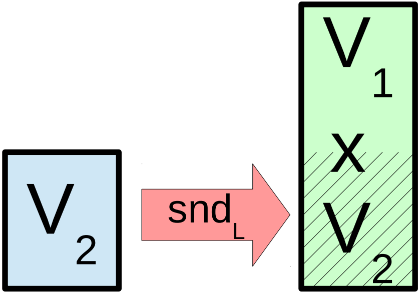
Lens sum

- $X \oplus Y$ parallel composes two independent lenses
- ***fst*** $: V_1 \Longrightarrow V_1 \times V_2$ injects the first component



Lens sum

- $X \oplus Y$ parallel composes two independent lenses
- $\textit{\textbf{fst}} : V_1 \Longrightarrow V_1 \times V_2$ injects the first component
- $\textit{\textbf{snd}} : V_2 \Longrightarrow V_1 \times V_2$ injects the second component



Lens sum

- $X \oplus Y$ parallel composes two independent lenses
- $\textbf{\textit{fst}} : V_1 \Longrightarrow V_1 \times V_2$ injects the first component
- $\textbf{\textit{snd}} : V_2 \Longrightarrow V_1 \times V_2$ injects the second component

$$
\begin{array}{rcll}
X \oplus \mathbf{0} & \approx & X & \\
X \oplus Y & \approx & Y \oplus X & X \bowtie Y \\
(X \oplus Y) \oplus Z & \approx & X \oplus (Y \oplus Z) & X, Y, Z \bowtie Y, Z, X \\
X & \preceq & X \oplus Y & \\
\textbf{\textit{fst}} \mathbin{;} (X \oplus Y) & = & X & X \bowtie Y \\
\textbf{\textit{snd}} \mathbin{;} (X \oplus Y) & = & Y & X \bowtie Y \\
\textbf{\textit{fst}} & \bowtie & \textbf{\textit{snd}} & \\
\textbf{\textit{fst}} \oplus \textbf{\textit{snd}} & = & \mathbf{1} &
\end{array}
$$

Lens sum

- $X \oplus Y$ parallel composes two independent lenses
- $\textit{fst} : V_1 \Longrightarrow V_1 \times V_2$ injects the first component
- $\textit{snd} : V_2 \Longrightarrow V_1 \times V_2$ injects the second component

$$\begin{array}{rcll}
X \oplus \mathbf{0} & \approx & X & \\
X \oplus Y & \approx & Y \oplus X & X \bowtie Y \\
(X \oplus Y) \oplus Z & \approx & X \oplus (Y \oplus Z) & X, Y, Z \bowtie Y, Z, X \\
X & \preceq & X \oplus Y & \\
\textit{fst} \,;\, (X \oplus Y) & = & X & X \bowtie Y \\
\textit{snd} \,;\, (X \oplus Y) & = & Y & X \bowtie Y \\
\textit{fst} & \bowtie & \textit{snd} & \\
\textit{fst} \oplus \textit{snd} & = & \mathbf{1} &
\end{array}$$

- similarity with separation algebra axioms

Summary

- lens composition: $X \fatsemi Y$
- identity lens: $\mathbf{1}$
- unit lens: $\mathbf{0}$
- sublens: $X \preceq Y$
- lens equivalence: $X \approx Y$
- lens sum: $P \oplus Q$
- first, second lens: ***fst***, ***snd***

Table of Contents

Alphabetised predicates

- recall our basic predicate model $\mathbb{P}\,\mathcal{S}$
- we augment this with lenses to model the variables
- also variable sets using $\fatsemi$ for $\cup$
- alphabets are modelled as Isabelle types ($\mathcal{S} = \alpha$)
- based on previous embedding of the UTP (Feliachi, 2010)

Alphabetised predicates

- recall our basic predicate model $\mathbb{P}\,\mathcal{S}$
- we augment this with lenses to model the variables
- also variable sets using $⨾$ for $\cup$
- alphabets are modelled as Isabelle types ($\mathcal{S} = \alpha$)
- based on previous embedding of the UTP (Feliachi, 2010)

- expressions: $(\tau, \alpha)\,\mathsf{uexpr} \triangleq (\alpha \Rightarrow \tau)$
- predicates: $\alpha\,\mathsf{upred} \triangleq (\mathsf{bool}, \alpha)\,\mathsf{uexpr}$
- relations: $(\alpha, \beta)\,\mathsf{urel} \triangleq (\alpha \times \beta)\,\mathsf{upred}$
- variables: $(\tau, \alpha)\,\mathsf{uvar} \triangleq (\tau \Longrightarrow \alpha)$

Alphabetised predicates

- recall our basic predicate model $\mathbb{P}\,\mathcal{S}$
- we augment this with lenses to model the variables
- also variable sets using $\fatsemi$ for $\cup$
- alphabets are modelled as Isabelle types ($\mathcal{S} = \alpha$)
- based on previous embedding of the UTP (Feliachi, 2010)

- expressions: $(\tau, \alpha)\,\mathsf{uexpr} \triangleq (\alpha \Rightarrow \tau)$
- predicates: $\alpha\,\mathsf{upred} \triangleq (\mathsf{bool}, \alpha)\,\mathsf{uexpr}$
- relations: $(\alpha, \beta)\,\mathsf{urel} \triangleq (\alpha \times \beta)\,\mathsf{upred}$
- variables: $(\tau, \alpha)\,\mathsf{uvar} \triangleq (\tau \Longrightarrow \alpha)$

- predicate operators created by lifting Isabelle/HOL equivalents

$$\begin{aligned} [\![\mathbf{true}]\!] &\triangleq \lambda\, s.\, \mathit{True} \\ [\![P \wedge Q]\!] &\triangleq \lambda\, s.\, [\![P]\!](s) \wedge [\![Q]\!](s) \end{aligned}$$

UTP variables

- lens operations model variable manipulations:

$$
\begin{array}{rcl}
x = y & \leadsto & x \approx y \\
x \neq y & \leadsto & x \bowtie y \\
x & \leadsto & x \,;\, \textbf{\textit{fst}} \\
x' & \leadsto & x \,;\, \textbf{\textit{snd}}
\end{array}
$$

UTP variables

- lens operations model variable manipulations:

$$
\begin{aligned}
x = y &\quad\rightsquigarrow\quad x \approx y \\
x \neq y &\quad\rightsquigarrow\quad x \bowtie y \\
x &\quad\rightsquigarrow\quad x \,; \textbf{\textit{fst}} \\
x' &\quad\rightsquigarrow\quad x \,; \textbf{\textit{snd}}
\end{aligned}
$$

- core predicate variable constructs:

$$
\begin{aligned}
[\![x]\!] &\triangleq \lambda\, s.\, \textit{get}_x\, s \\
[\![\exists\, x \bullet P]\!] &\triangleq (\lambda\, s.\, \exists\, v. P(\textit{put}_x\, s\, v)) \\
[\![\forall\, x \bullet P]\!] &\triangleq (\lambda\, s.\, \forall\, v. P(\textit{put}_x\, s\, v))
\end{aligned}
$$

Quantifier laws

Theorem (Cylindric Algebra)

$$(\exists\, x \bullet \textit{\textbf{false}}) \Leftrightarrow \textit{\textbf{false}}$$

$$P \Rightarrow (\exists\, x \bullet P)$$

$$(\exists\, x \bullet (P \wedge (\exists\, x \bullet Q))) \Leftrightarrow ((\exists\, x \bullet P) \wedge (\exists\, x \bullet Q))$$

$$(\exists\, x \bullet \exists\, y \bullet P) \Leftrightarrow (\exists\, y \bullet \exists\, x \bullet P)$$

$$(x = x) \Leftrightarrow \textit{\textbf{true}}$$

$$(y = z) \Leftrightarrow (\exists\, x \bullet y = x \wedge x = z) \qquad x \bowtie y, x \bowtie z$$

$$\textit{\textbf{false}} \Leftrightarrow \left(\begin{array}{l} (\exists\, x \bullet x = y \wedge P) \wedge \\ (\exists\, x \bullet x = y \wedge \neg P) \end{array} \right) \qquad x \bowtie y$$

Quantifier laws

Theorem (Cylindric Algebra)

$$(\exists\, x \bullet \textit{\textbf{false}}) \Leftrightarrow \textit{\textbf{false}}$$

$$P \Rightarrow (\exists\, x \bullet P)$$

$$(\exists\, x \bullet (P \wedge (\exists\, x \bullet Q))) \Leftrightarrow ((\exists\, x \bullet P) \wedge (\exists\, x \bullet Q))$$

$$(\exists\, x \bullet \exists\, y \bullet P) \Leftrightarrow (\exists\, y \bullet \exists\, x \bullet P)$$

$$(x = x) \Leftrightarrow \textit{\textbf{true}}$$

$$(y = z) \Leftrightarrow (\exists\, x \bullet y = x \wedge x = z) \qquad x \bowtie y, x \bowtie z$$

$$\textit{\textbf{false}} \Leftrightarrow \left(\begin{array}{l} (\exists\, x \bullet x = y \wedge P) \wedge \\ (\exists\, x \bullet x = y \wedge \neg P) \end{array} \right) \qquad x \bowtie y$$

Theorem (Other quantifier laws)

$$(\exists\, A \oplus B \bullet P) = (\exists\, A \bullet \exists\, B \bullet P)$$

$$(\exists\, B \bullet \exists\, A \bullet P) = (\exists\, A \bullet P) \qquad B \preceq A$$

$$(\exists\, x \bullet P) = (\exists\, y \bullet Q) \qquad x \approx y$$

Fresh variables

- unrestriction: semantic characterisation of fresh variables

Fresh variables

- unrestriction: semantic characterisation of fresh variables
- $x \sharp P$ if P's observations are independent of lens x

$$
\begin{aligned}
x \sharp P &\Leftrightarrow (\forall\, s \in P \bullet \forall\, v : V \bullet \mathsf{put}_x\, s\, v \in P) \\
&\Leftrightarrow P = (\exists\, x . P)
\end{aligned}
$$

Fresh variables

- unrestriction: semantic characterisation of fresh variables
- $x \sharp P$ if P's observations are independent of lens x

$$
\begin{aligned}
x \sharp P \;&\Leftrightarrow\; (\forall\, s \in P \bullet \forall\, v : V \bullet \mathsf{put}_x\, s\, v \in P) \\
&\Leftrightarrow\; P = (\exists\, x . P)
\end{aligned}
$$

- proven unrestriction laws:

$$
\frac{-}{\mathbf{0} \sharp P} \qquad \frac{x \preceq y \qquad y \sharp P}{x \sharp P} \qquad \frac{x \sharp P \qquad y \sharp P \qquad x \bowtie y}{(x \oplus y) \sharp P} \qquad \frac{x \bowtie y}{x \sharp y}
$$

$$
\frac{-}{x \sharp \mathbf{true}} \qquad \frac{-}{x \sharp \mathbf{false}} \qquad \frac{x \sharp P \qquad x \sharp Q}{x \sharp P \wedge Q} \qquad \frac{x \sharp P}{x \sharp \neg P} \qquad \frac{x \in \textsf{mwb-lens}}{x \sharp (\exists\, x \bullet P)} \qquad \frac{x \bowtie y \qquad x \sharp P}{x \sharp (\exists\, y \bullet P)}
$$

$$
\frac{x \sharp P}{x \sharp (P ; Q)} \qquad \frac{x' \sharp Q}{x' \sharp (P ; Q)}
$$

Substitution

- a substitution ($\sigma : \alpha\,\mathsf{usubst}$) is a function on state space α

$$\alpha\,\mathsf{usubst} \triangleq \alpha \Rightarrow \alpha$$

- identity substitution: $\mathsf{id} \triangleq \lambda\, x.x$
- update: $\sigma(x \mapsto_s v)$ for $x : (\tau, \alpha)\,\mathsf{uvar}$, $v : (\tau, \alpha)\,\mathsf{uexpr}$
- substitution application:

$$\begin{aligned} \sigma \dagger P &\triangleq \sigma[P] \\ P[v_1 \cdots v_n / x_1 \cdots x_n] &\triangleq [x_1 \mapsto v_1 \cdots x_n \mapsto v_n] \dagger P \end{aligned}$$

- some proven laws:

$$\frac{x \in \mathsf{mwb\text{-}lens}}{(\exists\, x \bullet P)[v/x] = (\exists\, x \bullet P)} \qquad \frac{x \bowtie y, y \sharp v}{(\exists\, y \bullet P)[v/x] = (\exists\, y \bullet P[v/x])}$$

Laws of programming

Theorem (Unital quantale)

UTP relations form a unital quantale and thus a Kleene algebra (Armstrong, 2015)

Laws of programming

Theorem (Unital quantale)

UTP relations form a unital quantale and thus a Kleene algebra (Armstrong, 2015)

Theorem (Assignment laws)

$$x := e \,;\, P \;=\; P[e/x]$$

$$x := e \,;\, x := f \;=\; x := f \qquad x \sharp f$$

$$x := e \,;\, y := f \;=\; y := f \,;\, x := e \qquad x \bowtie y, x \sharp f, y \sharp e$$

$$x := e \,;\, (P \lhd b \rhd Q) \;=\; (x := e \,;\, P) \lhd b[e/x] \rhd (x := e \,;\, Q) \qquad \mathbf{1}' \sharp b$$

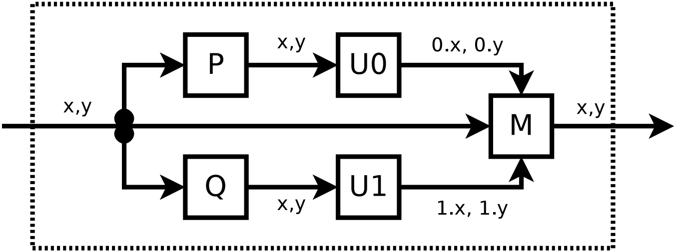
Table of Contents

Parallel by merge

- $P \parallel_M Q$ – general scheme for parallelism with merge M

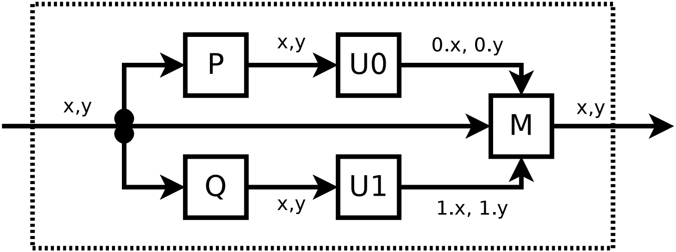
Parallel by merge

- $P \parallel_M Q$ – general scheme for parallelism with merge M



Parallel by merge

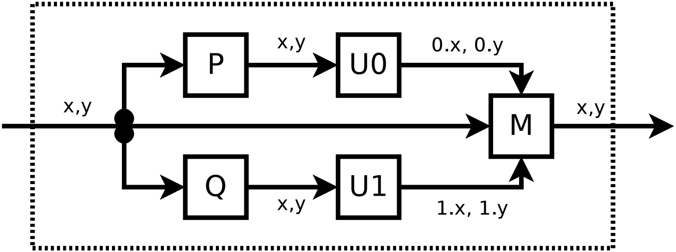
- $P \parallel_M Q$ – general scheme for parallelism with merge M



- can use lenses to express division of state space (A)
- i.e. $B_1 \oplus B_2 \approx A$ for disjoint alphabets $B_1 \bowtie B_2$

Parallel by merge

- $P \parallel_M Q$ – general scheme for parallelism with merge M



- can use lenses to express division of state space (A)
- i.e. $B_1 \oplus B_2 \approx A$ for disjoint alphabets $B_1 \bowtie B_2$
- merge relation type: $M : (A \times B_1 \times B_2, A)\,\textsf{urel}$

Differential equations

- hybrid systems combine computation + continuous dynamics
- we have developed a UTP theory of hybrid relations
- divide state into discrete (x, x') and continuous $(\underline{x})$

$$\underline{x} = \mathcal{F}(\underline{x}, \underline{\dot{x}}, x, y)$$

- $\underline{x}$ is a vector of real variables $(\mathbb{R}^n)$
- use lenses to focus on particular continuous variables
- allows to change how dynamics described

Table of Contents

Conclusion

- presented a general scheme for modelling state
- variables become entities in a larger abstract space
- through a theory of lenses and associated algebra
- have generically proved many of the laws of programming
- lenses can unify a variety of state-space models
- are there other applications of the theory?
- need to explore links (e.g. Back's variable calculus)

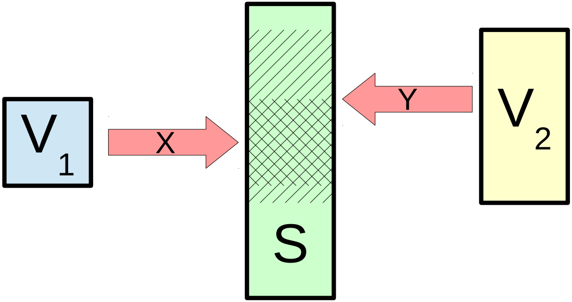
- Isabelle/UTP:
  github.com/isabelle-utp/utp-main
- Lenses:
  ../utils/Lenses.thy

Lens quotient

- $X /_L Y$ the dual operation of $X \fatsemi Y$
- assuming $X \preceq Y$, chop Y off from the end of X

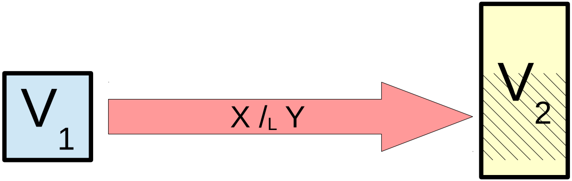
Lens quotient

- $X /_L Y$ the dual operation of $X \fatsemi Y$
- assuming $X \preceq Y$, chop Y off from the end of X



Lens quotient

- $X \mathbin{/_L} Y$ the dual operation of $X \fatsemi Y$
- assuming $X \preceq Y$, chop Y off from the end of X



Lens quotient

- $X /_L Y$ the dual operation of $X \fatsemi Y$
- assuming $X \preceq Y$, chop Y off from the end of X

$$
\begin{aligned}
(X /_L Y) \fatsemi Y &= X \\
(X \fatsemi Y) /_L Y &= X \\
(X /_L X) &= \mathbf{1} \\
(X /_L \mathbf{1}) &= X \\
(\mathbf{0} /_L X) &= \mathbf{0} \\
(X \oplus Y) /_L Z &= (X /_L Z) \oplus (Y /_L Z)
\end{aligned}
$$

Alphabet extrusion and restriction

- describe the extension and contraction of the state space

$$\_\oplus_p\_ \;:\; \beta\,\mathsf{upred} \Rightarrow (\beta \Longrightarrow \alpha) \Rightarrow \alpha\,\mathsf{upred}$$

$$P \oplus_p A \;=\; \{s \mid \mathsf{get}_A\; s \in P\}$$

$$\_\upharpoonright_p\_ \;:\; \alpha\,\mathsf{upred} \Rightarrow (\beta \Longrightarrow \alpha) \Rightarrow \beta\,\mathsf{upred}$$

$$P \upharpoonright_p A \;=\; \{s \mid \mathit{create}_A\; s \in P\}$$

- distributes through most predicate operators

$$\begin{aligned} P \oplus_p \mathbf{1} &= P \upharpoonright_p \mathbf{1} = P \\ \mathbf{true} \oplus_p A &= \mathbf{true} \\ (P \oplus_p A) \upharpoonright_p A &= P \end{aligned}$$

$$\frac{A \in \mathsf{mwb\text{-}lens},\; (A \oplus B) \in \mathsf{bij\text{-}lens},\; A \bowtie B,\; B \sharp P}{(P \upharpoonright_p A) \oplus_p A = P}$$