14 Foundation of Programming Languages and Software Engineering: *Abstract Data Types*

Summer Term 2010

Robert Elsässer

Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Abstract data types

- Problem-specific concepts
  - Search trees
  - Lists
  - Queues
  - . . .
- Implementations of these concepts may have different characteristics:
  - Memory usage
  - Efficiency
- Implementations should be exchangeable
- Abstract over the concepts, use ADTs!
  - Functional specification
  - Implementation independent
  - Different implementations of a single ADT are possible

ADTs are special signatures

Definition

Let Σ be a signature.

- A Σ-identity is a pair $s \approx t \in T(\Sigma, X) \times T(\Sigma, X)$.
- An ADT is a pair $(\Sigma, \mathcal{E})$ where
  - Σ is a signature,
  - $\mathcal{E} \subseteq T(\Sigma, X) \times T(\Sigma, X)$ is a set of Σ-identities.

Examples

An ADT for natural numbers

$$\Sigma_{nat} = \{\mathtt{zero}^{(0)}, \mathtt{succ}^{(1)}\}$$
$$\mathcal{E}_{nat} = \emptyset$$

An ADT for integers

$$\Sigma_{int} = \{\mathtt{zero}^{(0)}, \mathtt{pred}^{(1)}, \mathtt{succ}^{(1)}\}$$
$$\begin{aligned}\mathcal{E}_{int} = \{&\mathtt{pred}(\mathtt{succ}(x)) = x,\\ &\mathtt{succ}(\mathtt{pred}(x)) = x\}\end{aligned}$$

Datatypes are Σ-Algebras

Definition

- A datatype is a Σ-algebra $\mathcal{D}$.
- A datatype $\mathcal{D}$ implements the ADT $(\Sigma, \mathcal{E})$ iff every identity $s \approx t \in \mathcal{E}$ is valid in $\mathcal{D}$.
  (Note: We shall refine this definition later.)
- An identity $s \approx t$ is valid in a Σ-algebra $\mathcal{A} = (A, \alpha)$ iff $\hat{\mathcal{J}}(s) = \hat{\mathcal{J}}(t)$ for all variable assignments $\mathcal{J} : X \to A$.

Implementations of the ADT for Naturals

Implementation 1

- $\mathcal{D}nat_1 = (\mathbb{N}, \alpha_1)$, $\alpha_1(\texttt{zero}) = 0$, $\alpha_1(\texttt{succ})(x) = x + 1$.
- No identities, so all are valid.
- The function $\widehat{\alpha}_1$ is bijective.

Implementations of the ADT for Naturals

Implementation 1

- $\mathcal{D}nat_1 = (\mathbb{N}, \alpha_1)$, $\alpha_1(\mathtt{zero}) = 0$, $\alpha_1(\mathtt{succ})(x) = x + 1$.
- No identities, so all are valid.
- The function $\widehat{\alpha}_1$ is bijective.

Implementation 2

- $\mathcal{D}nat_2 = (\{0, 1, 2, 3\}, \alpha_2)$, $\alpha_2(\mathtt{zero}) = 0$, $\alpha_2(\mathtt{succ})(x) = (x + 1) \mod 4$.
- No identities, so all are valid.
- The function $\widehat{\alpha}_2$ is not injective (but surjective). $\widehat{\alpha}_2(\mathtt{zero}) = 0 = \widehat{\alpha}_2(\mathtt{succ}(\mathtt{succ}(\mathtt{succ}(\mathtt{succ}(\mathtt{zero})))))$

Implementations of the ADT for Integers (1)

Implementation 1

- $\mathcal{D}int_1 = (\mathbb{Z}, \beta_1)$, $\begin{aligned} \beta_1(\mathtt{zero})() &= 0 \\ \beta_1(\mathtt{succ})(x) &= x + 1 \\ \beta_1(\mathtt{pred})(x) &= x - 1 \end{aligned}$
- For arbitrary $\mathcal{J} : \{x\} \to \mathbb{Z}$ we have
  $\hat{\mathcal{J}}(\mathtt{pred}(\mathtt{succ}(x))) = (\mathcal{J}(x) + 1) - 1 = \hat{\mathcal{J}}(x)$
  $\hat{\mathcal{J}}(\mathtt{succ}(\mathtt{pred}(x))) = (\mathcal{J}(x) - 1) + 1 = \hat{\mathcal{J}}(x)$
- $\widehat{\beta_1}$ is surjective but not injective. Consider
  $\widehat{\beta_1}(\mathtt{zero}) = 0 = \widehat{\beta_1}(\mathtt{succ}(\mathtt{pred}(\mathtt{zero})))$

Implementations of the ADT for Integers (2)

Implementation 2

- $\mathcal{D}int_2 = (\{0, 1, 2, 3\}, \beta_2)$,
  $$\beta_2(\texttt{zero})() \quad = 0$$
  $$\beta_2(\texttt{succ})(x) = x + 1 \mod 4$$
  $$\beta_2(\texttt{pred})(x) = \begin{cases} 3 & \text{if } x = 0, \\ x - 1 & \text{otherwise.} \end{cases}$$
- For arbitrary $\mathcal{J} : \{x\} \to \mathbb{Z}$ we have
  $$\hat{\mathcal{J}}(\texttt{pred}(\texttt{succ}(x))) = \hat{\mathcal{J}}(x)$$
  $$\hat{\mathcal{J}}(\texttt{succ}(\texttt{pred}(x))) = \hat{\mathcal{J}}(x)$$
- $\widehat{\beta_2}$ is surjective but not injective.

Implementations of the ADT for Integers (3)

A Non-implementation

- $\mathcal{D}int_3 = (\mathbb{N}, \beta_3)$

$$\begin{aligned} \beta_3(\mathtt{zero})() &= 0 \\ \beta_3(\mathtt{succ})(x) &= x + 1 \\ \beta_3(\mathtt{pred})(x) &= \begin{cases} x - 1 & x > 0 \\ 0 & x = 0 \end{cases} \end{aligned}$$

- Not an implementation:
  For $\mathcal{J} : X \to \mathbb{N}$ with $\mathcal{J}(x) = 0$ we have

$$\hat{\mathcal{J}}(\mathtt{succ}(\mathtt{pred}(x))) = 1 \neq 0 = \hat{\mathcal{J}}(x)$$

Fixing the Problems

- Want to rule out implementations such as $\mathcal{D}nat_2$ and $\mathcal{D}int_2$
- Definition of "implementation" is too weak
- Needed: restriction on function $\widehat{\alpha}$
  - $\widehat{\alpha}$ is not necessarily injective (see $\mathcal{D}int_1$)
  - Idea: $\widehat{\alpha}$ must be injective on the equivalence classes induced by the identities of an ADT.

Equivalence Classes

Definition

Suppose R is an equivalence relation on some set M.

- The set $[x]_R := \{y \in M \mid x\, R\, y\}$ is called the equivalence class of x.
- $y \in [x]_R$ is called a representative of $[x]_R$.
- The quotient of M with respect to R is the set of equivalence classes induced by R, written $M/_R := \{[x]_R \mid x \in M\}$.

Note: For equivalence classes $[x]_R$ and $[y]_R$ we have either $[x]_R = [y]_R$ or $[x]_R \cap [y]_R = \emptyset$.

Congruence Relations

Definition

Suppose Σ is a signature and let R be an equivalence relation on $T(\Sigma, X)$.

- R is a congruence relation iff R is closed under Σ-operations, i.e. $s\, R\, s'$ implies $f(t_1, \ldots, s, \ldots, t_n)\ R\ f(t_1, \ldots, s', \ldots, t_n)$ for any $n \geq 0$, $f \in \Sigma^{(n)}$, and $s, s', t_1, \ldots, t_n \in T(\Sigma, X)$.

Syntactic Quotient Algebras

Lemma

Let Σ be a signature and R be a congruence on $T(\Sigma, X)$. For all $n \geq 0$, $f \in \Sigma^{(n)}$, and $t_1, \ldots, t_n \in T(\Sigma, X)$, define α^R as follows:

$$\alpha^R(f)([t_1]_R, \ldots, [t_n]_R) = [f(t_1, \ldots, t_n)]_R$$

Then $(T(\Sigma, X)/_R, \alpha^R)$ is a Σ-algebra.

Proof. We need to show that α^R is well-defined. Suppose $n \geq 0$, $f \in \Sigma^{(n)}$, and $s_1, t_1, \ldots, s_n, t_n \in T(\Sigma, X)$. If $s_1 \; R \; t_1, \ldots, s_n \; R \; t_n$ then $f(s_1, \ldots, s_n) \; R \; f(t_1, \ldots, t_n)$. Hence, $[f(s_1, \ldots, s_n)]_R = [f(t_1, \ldots, t_n)]_R$ because R is a congruence.

Equational Theory

Definition

Let $(\Sigma, \mathcal{E})$ be an ADT. We define a relation $\approx_{\mathcal{E}}$ on $T(\Sigma, X)$ as the smallest relation such that

- $\approx_{\mathcal{E}}$ is a congruence relation;
- $\approx_{\mathcal{E}}$ contains $\mathcal{E}$, i.e. $s \approx t \in \mathcal{E}$ implies $s \approx_{\mathcal{E}} t$;
- $\approx_{\mathcal{E}}$ is closed under substitutions, i.e. $s \approx_{\mathcal{E}} t$ implies $\sigma(s) \approx_{\mathcal{E}} \sigma(t)$ for any substitution σ and all $s, t \in T(\Sigma, X)$.

Example

Congruence classes of $\approx_{\mathcal{E}_{int}}$

$$\Sigma_{int} = \{\mathtt{zero}^{(0)}, \mathtt{pred}^{(1)}, \mathtt{succ}^{(1)}\}$$

$$\begin{aligned}\mathcal{E}_{int} = \{&\mathtt{pred}(\mathtt{succ}(x)) = x,\\ &\mathtt{succ}(\mathtt{pred}(x)) = x\}\end{aligned}$$

$$\begin{aligned}[\mathtt{zero}]_{\approx_{\mathcal{E}_{int}}} = \{&\mathtt{zero},\\ &\mathtt{succ}(\mathtt{pred}(\mathtt{zero})),\\ &\mathtt{pred}(\mathtt{succ}(\mathtt{zero})),\\ &\mathtt{succ}(\mathtt{succ}(\mathtt{pred}(\mathtt{pred}(\mathtt{zero})))), \ldots\}\end{aligned}$$

Revised Definition for ADT Implementations

Definition

A datatype $\mathcal{D} = (M, \alpha)$ implements ADT $(\Sigma, \mathcal{E})$ with constructors $\Gamma \subseteq \Sigma$ if

- (M, α) is a Σ-algebra
- All identities from $\mathcal{E}$ are valid in M
- For all $s, t \in T(\Gamma, \emptyset)$: $s \approx_{\mathcal{E}} t$ iff $\widehat{\alpha}(s) = \widehat{\alpha}(t)$

Syntactic Implementations of ADTs

Theorem

Let $(\Sigma, \mathcal{E})$ be an ADT with constructors $\Gamma \subseteq \Sigma$. Then $\mathcal{D} = (T(\Sigma, \emptyset)/_{\approx_\mathcal{E}}, \alpha^{\approx_\mathcal{E}})$ is an implementation of $(\Sigma, \mathcal{E})$.

Proof. $\mathcal{D}$ is a Σ-algebra because $\approx_\mathcal{E}$ is a congruence.
An easy term induction shows that

$$\widehat{\alpha}^{\approx_\mathcal{E}}(t) = [t]_{\approx_\mathcal{E}} \tag{1}$$

holds for all $t \in T(\Sigma, \emptyset)$.
By using (1), we show for all $s, t \in T(\Gamma, \emptyset)$ that $s \approx_\mathcal{E} t$ iff $\widehat{\alpha}^{\approx_\mathcal{E}}(s) = \widehat{\alpha}^{\approx_\mathcal{E}}(t)$.

Proof (cont.)

We still need to establish the validity of the identities in $\mathcal{E}$. Suppose $\hat{\mathcal{J}}$ is an interpretation function. We define a substitution σ as follows:

$$\sigma(x) = \begin{cases} x & \text{if } x \text{ does not appear in } s \text{ or } t, \\ t & \text{otherwise, where } \mathcal{J}(x) = [t]_{\approx_{\mathcal{E}}} \end{cases}$$

By term induction, we then show for all $r \in T(\Sigma, X)$

$$\hat{\mathcal{J}}(r) = \widehat{\alpha}^{\approx_{\mathcal{E}}}(\sigma(r)) \tag{2}$$

From $s \approx t \in \mathcal{E}$, we get $s \approx_{\mathcal{E}} t$, so $\sigma(s) \approx_{\mathcal{E}} \sigma(t)$ holds. We finish to proof by calculating

$$\hat{\mathcal{J}}(s) \overset{(2)}{=} \widehat{\alpha}^{\approx_{\mathcal{E}}}(\sigma(s)) \overset{(1)}{=} [\sigma(s)]_{\approx_{\mathcal{E}}} = [\sigma(t)]_{\approx_{\mathcal{E}}} \overset{(1)}{=} \widehat{\alpha}^{\approx_{\mathcal{E}}}(\sigma(t)) \overset{(2)}{=} \hat{\mathcal{J}}(t)$$

Example

Nat as a constructor-based ADT (CADT)

CADT: $\Sigma = \{\mathrm{z}, \mathrm{s}\}, \mathcal{E} = \{\}, \Gamma = \Sigma$
Implementation: $(\mathbb{N}, \alpha_1)$ with $\alpha_1(\mathrm{z})() = 0$ and $\alpha_1(\mathrm{s})(x) = x + 1$

- $(\mathbb{N}, \alpha_1)$ is Σ-algebra
- No identities to check
- Since $\mathcal{E} = \emptyset$, $\approx_{\mathcal{E}}$ is $=$. Suppose $s, t \in T(\Gamma, \emptyset)$.
  - If $s = t$ then $\widehat{\alpha}_1(s) = \widehat{\alpha}_1(t)$
  - Suppose $s \neq t$. Then $s = \mathrm{s}^n(t)$ with $n > 0$. Hence, $\widehat{\alpha}_1(s) = \widehat{\alpha}_1(t) + n \neq \widehat{\alpha}_1(t)$.

Example

$\mathcal{D}int_2$ is not an implementation of the natural numbers

CADT: $\Sigma = \{\mathrm{z}, \mathrm{s}\}, \mathcal{E} = \{\}, \Gamma = \Sigma$
$(\{0, 1, 2, 3\}, \alpha_2)$ with $\alpha_2(\mathrm{z})() = 0$, $\alpha_2(\mathrm{s})(x) = (x + 1) \mod 4$
is not an implementation.

- $(\{0, 1, 2, 3\}, \alpha_2)$ is Σ-algebra
- No identities to check
- Since $\mathcal{E} = \emptyset$, $\approx_{\mathcal{E}}$ is $=$.
  We have $\mathrm{z} \neq \mathrm{s}^4(\mathrm{z})$ but $\widehat{\alpha}_2(\mathrm{z}) = 0 = \widehat{\alpha}_2(\mathrm{s}^4(\mathrm{z}))$.

Example

Alternative implementation of the natural numbers

CADT: $\Sigma = \{\mathrm{z}, \mathrm{s}\}, \mathcal{E} = \{\}, \Gamma = \Sigma$

Implementation: $(\{a\}^*, \alpha_3)$ with $\alpha_3(\mathrm{z})() = \epsilon$, $\alpha_3(\mathrm{s})(w) = aw$

- $(\{a\}^*, \alpha_3)$ is Σ-algebra
- No identities to check
- Since $\mathcal{E} = \emptyset$, $\approx_{\mathcal{E}}$ is $=$.
    - If $s = t$ then $\widehat{\alpha}_3(s) = \widehat{\alpha}_3(t)$
    - Suppose $s \neq t$. Then $s = \mathrm{s}^n(t)$ with $n > 0$. Hence, $\widehat{\alpha}_3(s) = \widehat{\alpha}_3(t)\underbrace{a \dots a}_{n} \neq \widehat{\alpha}_3(t)$.

Equivalence Classes for Terms Representing Integers

Suppose $\Gamma = \Sigma = \{\mathrm{z}, \mathrm{s}, \mathrm{p}\}$, $\mathcal{E} = \{\mathrm{s}(\mathrm{p}(x)) = x, \mathrm{p}(\mathrm{s}(x)) = x\}$
Question: What is $T(\Sigma, \emptyset)/_{\approx_\mathcal{E}}$?
Answer: Give a representative for every equivalence class.

Lemma

For every term $t \in T(\Sigma, \emptyset)$, exactly one of the following propositions holds

- A There exists $n > 0$ such that $t \in [\mathrm{s}^n(\mathrm{z})]_{\approx_\mathcal{E}}$.
- B $t \in [\mathrm{z}]_{\approx_\mathcal{E}}$.
- C There exists $n > 0$ such that $t \in [\mathrm{p}^n(\mathrm{z})]_{\approx_\mathcal{E}}$.

Proof

The proof is by term induction over t.

- $t = \mathrm{z}$ so B holds.
- Induction Step for $t = \mathrm{s}(t')$. By the IH, one of the following holds for t'.
  - A If $t' \approx_{\mathcal{E}} \mathrm{s}^{(n)}(\mathrm{z})$ for $n > 0$
    then $\mathrm{s}(t') \approx_{\mathcal{E}} \mathrm{s}(\mathrm{s}^{(n)}(\mathrm{z})) = \mathrm{s}^{(n+1)}(\mathrm{z})$.
    Since $n + 1 > 0$ we have case A.
  - B If $t' \approx_{\mathcal{E}} \mathrm{z}$ then $\mathrm{s}(t') \approx_{\mathcal{E}} \mathrm{s}(\mathrm{z})$.
    We have case A with $n = 1$.
  - C If $t' \approx_{\mathcal{E}} \mathrm{p}^{(n)}(\mathrm{z})$ for $n > 0$
    then $\mathrm{s}(t') \approx_{\mathcal{E}} \mathrm{s}(\mathrm{p}^{(n)}(\mathrm{z}))$.
    If $n = 1$ then $\mathrm{s}(\mathrm{p}(\mathrm{z})) \approx_{\mathcal{E}} \mathrm{z}$ so case B holds.
    If $n > 1$ then $\mathrm{s}(\mathrm{p}(\mathrm{p}^{(n-1)}(\mathrm{z}))) \approx_{\mathcal{E}} \mathrm{p}^{(n-1)}(\mathrm{z})$, so case C holds.
- Induction step for $\mathrm{p}(t)$ analogous.

Equivalence Classes for Terms Representing Integers

Lemma

Suppose $n > 0$, $m > 0$. Then we have

- $\mathrm{z} \not\approx_{\mathcal{E}} \mathrm{s}^n(\mathrm{z})$,
- $\mathrm{z} \not\approx_{\mathcal{E}} \mathrm{p}^n(\mathrm{z})$,
- $\mathrm{s}^n(\mathrm{z}) \not\approx_{\mathcal{E}} \mathrm{p}^m(\mathrm{z})$,
- $\mathrm{s}^n(\mathrm{z}) \not\approx_{\mathcal{E}} \mathrm{s}^m(\mathrm{z})$ provided $n \neq m$, and
- $\mathrm{p}^n(\mathrm{z}) \not\approx_{\mathcal{E}} \mathrm{p}^m(\mathrm{z})$ provided $n \neq m$.

If follows that

$$\{\mathrm{s}^n(\mathrm{z}) | n > 0\} \cup \{\mathrm{z}\} \cup \{\mathrm{p}^n(\mathrm{z}) | n > 0\}$$

is a set of representatives for $T(\Sigma, \emptyset)/_{\approx_{\mathcal{E}}}$.

Example

Integers as a CADT

CADT: $\Gamma = \Sigma = \{\mathrm{z}, \mathrm{s}, \mathrm{p}\}, \mathcal{E} = \{\mathrm{s}(\mathrm{p}(x)) = x, \mathrm{p}(\mathrm{s}(x)) = x\}$
Implementation: $(\mathbb{Z}, \alpha)$ with
$\alpha(z) = 0, \alpha(\mathrm{s})(x) = x + 1, \alpha(\mathrm{p})(x) = x - 1$

- $(\mathbb{Z}, \alpha)$ is a Σ-algebra
- All identities are valid (as seen before)
- An easy term induction shows for all $t \in T(\Sigma, \emptyset)$ that
  - if $t \approx_{\mathcal{E}} \mathrm{z}$ then $\widehat{\alpha}(t) = 0$,
  - if $t \approx_{\mathcal{E}} \mathrm{s}^n(\mathrm{z})$ then $\widehat{\alpha}(t) = n$, and
  - if $t \approx_{\mathcal{E}} \mathrm{p}^n(\mathrm{z})$ then $\widehat{\alpha}(t) = -n$.

  Hence, if $s \approx_{\mathcal{E}} t$ then $\widehat{\alpha}(s) = \widehat{\alpha}(t)$.
  Conversely, if $s \not\approx_{\mathcal{E}} t$ then $\widehat{\alpha}(s) \neq \widehat{\alpha}(t)$ because $\widehat{\alpha}$ maps different representatives to different integers.

Summary

Slogan

Calculating with ADT $=$
applying term operations $+$
determining set of representatives.

Richer ADTs

Definition (Linear Data Structure)

An ADT is called a linear data structure (LDS) iff there is an implementation with simple lists.

- LDS are aggregates, i.e. one element contains several elements of another sort.
- Types are now parameterized.
  - Examples: List(A), Array(A)
  - A is not a fixed type but rather a type parameter.
  - Compare with Java Generics: `List<A>`, `A[]`

Lists

Definition (Signature for Lists)

data type	$\mathsf{List}(A)$	
operations	`empty`	$: \to \mathsf{List}(A)$
	`cons`	$: A \times \mathsf{List}(A) \to \mathsf{List}(A)$
	`head`	$: \mathsf{List}(A) \to A$
	`tail`	$: \mathsf{List}(A) \to \mathsf{List}(A)$
	`empty?`	$: \mathsf{List}(A) \to \mathsf{Boolean}$
	`app`	$: \mathsf{List}(A) \times \mathsf{List}(A) \to \mathsf{List}(A)$
	`len`	$: \mathsf{List}(A) \to \mathsf{Nat}$

- Definition parameterized over A
- Constructors: `empty`, `cons`
- Definition uses more than one type: More general notion of signature and arity needed

Lists (cont.)

Definition (Identities for Lists)

identities

$$\mathtt{head}(\mathtt{cons}(a, l)) = a$$
$$\mathtt{tail}(\mathtt{cons}(a, l)) = l$$
$$\mathtt{empty?}(\mathtt{empty}) = \mathit{true}$$
$$\mathtt{empty?}(\mathtt{cons}(a, l)) = \mathit{false}$$
$$\mathtt{app}(\mathtt{empty}, v) = v$$
$$\mathtt{app}(\mathtt{cons}(a, l), v) = \mathtt{cons}(a, \mathtt{app}(l, v))$$
$$\mathtt{len}(\mathtt{empty}) = \mathtt{zero}$$
$$\mathtt{len}(\mathtt{cons}(a, l)) = \mathtt{succ}(\mathtt{len}(l))$$

Abstract Data Types
Heterogeneous Signatures

Definition

Let S be a set of sorts. A heterogeneous signature Σ is a set of function symbols where each $f \in \Sigma$ is associated with an arity $s \to s'$ where $s \in S^*$ and $s' \in S$.

Examples

- Arity of `empty`: $\epsilon \to \mathsf{List}(A)$
- Arity of `cons`: $(A, \mathsf{List}(A)) \to \mathsf{List}(A)$

Previous definitions need to be generalized as well:

- An algebra has different carrier sets for every sort.
- Terms must respect the sorts associated with a function symbol to rule out illegal terms such as $\mathtt{cons}(1, 1)$.
- Generalize congruence relation

Dealing with Partial Operations

- $\mathtt{head}$ and $\mathtt{tail}$ are partial operations:
  - $\mathtt{head}(\mathtt{empty}) = ??$
  - $\mathtt{tail}(\mathtt{empty}) = ??$
- One possible solution: Introduce a distinguished element $\bot_s$ for every sort s
  - $\mathtt{head}(\mathtt{empty}) = \bot_A$
  - $\mathtt{tail}(\mathtt{empty}) = \bot_{\mathsf{List}(A)}$
- All operations are strict in $\bot_s$, i.e. if one argument is $\bot_s$ the result is $\bot_{s'}$
  - $\mathtt{head}(\bot_{\mathsf{List}(A)}) = \bot_A$
  - $\mathtt{tail}(\bot_{\mathsf{List}(A)}) = \bot_{\mathsf{List}(A)}$
  - $\mathtt{empty?}(\bot_{\mathsf{List}(A)}) = \bot_{\mathsf{Boolean}}$
  - $\mathtt{len}(\bot_{\mathsf{List}(A)}) = \bot_{\mathsf{Nat}}$
  - $\ldots$

Search for Representatives

Proposition

Let t be a term of type List(A) without variables. Then one of the following holds:

- $t \approx_{\mathcal{E}} t'$ and $t' \in T(\Gamma, \emptyset)$ where $\Gamma = \{\mathtt{empty}, \mathtt{cons}\}$.
- $t \approx_{\mathcal{E}} \perp_{\text{List}(A)}$

Proof. The proof is by induction on t.

- Case $t = \mathtt{empty} \in T(\Gamma, \emptyset)$. Trivial.
- Case $t = \mathtt{cons}(a, s)$ with $s \approx_{\mathcal{E}} s'$ and $s' \in T(\Gamma, \emptyset)$.
  Hence, $\mathtt{cons}(a, s) \approx_{\mathcal{E}} \mathtt{cons}(a, s') \in T(\Gamma, \emptyset)$.
- Case $t = \mathtt{head}(s)$ so t does not have type List(A).
- Case $t = \mathtt{tail}(s)$ with $s \approx_{\mathcal{E}} s'$ and $s' \in T(\Gamma, \emptyset)$.
  If $s' = \mathtt{empty}$ then $t \approx_{\mathcal{E}} \perp_{\text{List}(A)}$.
  If $s' = \mathtt{cons}(a, s'')$ then $t \approx_{\mathcal{E}} s'' \in T(\Gamma, \emptyset)$.

Proof (cont.)

- Case $t = \mathtt{empty?}(s)$ but then t does not have type List(A).
- Case $t = \mathtt{app}(s_1, s_2)$ with $s_1 \approx_{\mathcal{E}} s_1'$ and $s_2 \approx_{\mathcal{E}} s_2'$ and $s_1', s_2' \in T(\Gamma, \emptyset)$.
    - If $s_1' = \mathtt{empty}$ then $t \approx_{\mathcal{E}} \mathtt{app}(\mathtt{empty}, s_2) \approx_{\mathcal{E}} s_2 \approx_{\mathcal{E}} s_2'$.
    - If $s_1' = \mathtt{cons}(a, s_1'')$ then
    $$\begin{aligned} t = \mathtt{app}(s_1, s_2) &\approx_{\mathcal{E}} \mathtt{app}(\mathtt{cons}(a, s_1''), s_2) \\ &\approx_{\mathcal{E}} \mathtt{cons}(a, \mathtt{app}(s_1'', s_2)) \end{aligned}$$
    By the IH, we have $\mathtt{app}(s_1'', s_2) \approx_{\mathcal{E}} s'$ with $s' \in T(\Gamma, \emptyset)$.
- Case $t = \mathtt{len}(s)$ but then t does not have type List(A).

Another Example

Sequences

- Also know as Array or Vector
- Parameterized over type of elements
- Fixed number of elements
- Direct access to elements (constant time)

Arrays

Definition (Signature for Arrays)

data type	$\mathsf{Array}(A)$
operations	$\texttt{new} \quad : \mathsf{Nat} \times A \rightarrow \mathsf{Array}(A)$
	$\texttt{update} : \mathsf{Array}(A) \times \mathsf{Nat} \times A \rightarrow \mathsf{Array}(A)$
	$\texttt{get} \quad : \mathsf{Array}(A) \times \mathsf{Nat} \rightarrow A$
	$\texttt{len} \quad : \mathsf{Array}(A) \rightarrow \mathsf{Nat}$

- Functional arrays
- In imperative languages: `update` operation changes the array

Arrays (cont.)

Definition (Identities for Arrays)

<u>identities</u>

(i.1)	$\mathtt{get}(\mathtt{new}(n, x), i) = x$	**if** $i < n$
(i.2)	$\mathtt{get}(\mathtt{update}(a, i, x), i) = x$	**if** $i < \mathtt{len}(a)$
(i.3)	$\mathtt{get}(\mathtt{update}(a, j, x), i) = \mathtt{get}(a, i)$	**if** $i \neq j$
(i.4)	$\mathtt{update}(\mathtt{update}(a, i, x), i, y) = \mathtt{update}(a, i, y)$	
(i.5)	$\mathtt{update}(\mathtt{update}(a, j, x), i, y) = \mathtt{update}(\mathtt{update}(a, i, y), j, x)$	**if** $i \neq j$
(i.6)	$\mathtt{update}(\mathtt{new}(n, x), i, x) = \mathtt{new}(n, x)$	**if** $i < n$
(i.7)	$\mathtt{len}(\mathtt{new}(n, x)) = n$	
(i.8)	$\mathtt{len}(\mathtt{update}(a, i, x)) = \mathtt{len}(a)$	

New: Conditional identities

Examples

Calculating with representatives

Suppose the carrier set of A is $\{a, b, c\}$.

1. $\mathtt{get}(\mathtt{new}(10, a), 5) \overset{(i.1)}{\approx_{\mathcal{E}}} a$
2. $\mathtt{get}(\mathtt{new}(10, a), 11) \approx_{\mathcal{E}} \perp_A$
3. $\mathtt{get}(\mathtt{update}(\mathtt{update}(\mathtt{new}(10, a), 5, b), 0, c), 5)$
   $\overset{(i.3)}{\approx_{\mathcal{E}}} \mathtt{get}(\mathtt{update}(\mathtt{new}(10, a), 5, b), 5) \overset{(i.2)}{\approx_{\mathcal{E}}} b$
4. $\mathtt{update}(\mathtt{update}(\mathtt{new}(10, a), 5, b), 5, a)$
   $\overset{(i.4)}{\approx_{\mathcal{E}}} \mathtt{update}(\mathtt{new}(10, a), 5, a) \overset{(i.6)}{\approx_{\mathcal{E}}} \mathtt{new}(10, a)$
5. $\mathtt{get}(\mathtt{update}(\mathtt{new}(0, a), 1, a), 1) \approx_{\mathcal{E}} \perp_A$
   (i.2) is not applicable because
   $\mathtt{len}(\mathtt{update}(\mathtt{new}(0, a), 1, a)) \approx_{\mathcal{E}} 0$.