Abstract Data Types

We have learned about different datastructures, e.g. for dictionaries:
- Search trees
- Lists
- Tables with hashing

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 $\Sigma$ be a signature.

- A $\Sigma$-identity is a pair $(s, t) \in T(\Sigma, X) \times T(\Sigma, X)$. We write a $\Sigma$-identity as $s \approx t$ for emphasis.
- An ADT is a pair $(\Sigma, E)$ where
  - $\Sigma$ is a signature,
  - $E \subseteq T(\Sigma, X) \times T(\Sigma, X)$ is a set of $\Sigma$-identities.

Examples

An ADT for natural numbers

$\Sigma_{nat} = \{\text{zero}^{(0)}, \text{succ}^{(1)}\}$
$E_{nat} = \emptyset$

An ADT for integers

$\Sigma_{int} = \{\text{zero}^{(0)}, \text{pred}^{(1)}, \text{succ}^{(1)}\}$
$E_{int} = \{\text{pred}(\text{succ}(x)) \approx x, \text{succ}(\text{pred}(x)) \approx x\}$
Datatypes are $\Sigma$-Algebras

**Definition**

- An identity $s \approx t$ is valid in a $\Sigma$-algebra $A = (A, J)$ iff $J_\alpha(s) = J_\alpha(t)$ for all variable assignments $\alpha : X \to A$.
- A datatype is a $\Sigma$-algebra $D$.
- A datatype $D$ implements the ADT $(\Sigma, E)$ iff every identity $s \approx t \in E$ is valid in $D$.

(Note: We shall refine this definition later.)

Implementations of the ADT for Naturals

**Implementation 1**

$Dnat' = (\mathbb{N}, J')$,
- $J'(zero) = 0$,
- $J'(succ(x)) = x + 1$,
- $J'(pred(x)) = x - 1$

For arbitrary $\alpha : \{x\} \to \mathbb{N}$ we have
- $J'_\alpha(pred(succ(x))) = (\alpha(x) + 1) - 1 = J'_\alpha(x)$
- $J'_\alpha(succ(pred(x))) = (\alpha(x) - 1) + 1 = J'_\alpha(x)$

$J'$ is surjective but not injective. Consider $J'(zero) = 0 = J'(succ(pred(zero)))$.

No identities, so all are valid.

The function $J'$ is bijective.

**Implementation 2**

$Dnat'' = (\{0, 1, 2, 3\}, J'')$,
- $J''(zero) = 0$,
- $J''(succ(x)) = (x + 1) \mod 4$,
- $J''(pred(x)) = x \mod 4$

For arbitrary $\alpha : \{x\} \to \mathbb{N}$ we have
- $J''_\alpha(pred(succ(x))) = J''_\alpha(x)$
- $J''_\alpha(succ(pred(x))) = J''_\alpha(x)$

$J''$ is surjective but not injective.

Implementations of the ADT for Integers (1)

**Implementation 1**

$Dint' = (\mathbb{Z}, J')$,
- $J'(zero)() = 0$,
- $J'(succ(x)) = x + 1$,
- $J'(pred(x)) = x - 1$

For arbitrary $\alpha : \{x\} \to \mathbb{Z}$ we have
- $J'_\alpha(pred(succ(x))) = (\alpha(x) + 1) - 1 = J'_\alpha(x)$
- $J'_\alpha(succ(pred(x))) = (\alpha(x) - 1) + 1 = J'_\alpha(x)$

$J'$ is surjective but not injective. Consider $J'(zero) = 0 = J'(succ(pred(zero)))$.

Implementations of the ADT for Integers (2)

**Implementation 2**

$Dint'' = (\{0, 1, 2, 3\}, J'')$,
- $J''(zero)() = 0$,
- $J''(succ(x)) = (x + 1) \mod 4$,
- $J''(pred(x)) = x \mod 4$

For arbitrary $\alpha : \{x\} \to \mathbb{N}$ we have
- $J''_\alpha(pred(succ(x))) = J''_\alpha(x)$
- $J''_\alpha(succ(pred(x))) = J''_\alpha(x)$

$J''$ is surjective but not injective.
Implementations of the ADT for Integers (3)

A Non-implementation

- $\text{Dint}'' = (\mathbb{N}, J'')$
  - $J''(\text{zero}()) = 0$
  - $J''(\text{succ})(x) = x + 1$
  - $J''(\text{pred})(x) = \begin{cases} x - 1 & x > 0 \\ 0 & x = 0 \end{cases}$

- Not an implementation:
  For $\alpha : X \to \mathbb{N}$ with $\alpha(x) = 0$ we have
  $$J_\alpha(\text{succ}(\text{pred}(x))) = 1 \neq 0 = J_\alpha(x)$$

Fixing the Problems

- Want to rule out implementations such as $\text{Dnat}'$ and $\text{Dint}'$
- Definition of “implementation” is too weak
- Needed: restriction on function $J$
  - $J$ is not necessarily injective (see $\text{Dint}'$)
  - Idea: $J$ must be injective on the equivalence classes induced by the identities of an ADT.
  In other words: $J$ should make those terms equal that are equal according to the identities of the ADT, but not more!

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 $\Sigma$ is a signature and let $R$ be an equivalence relation on $T(\Sigma, X)$.
- $R$ is a congruence relation iff $R$ is closed under $\Sigma$-operations, i.e. $t_i R t'_i$ implies $f(t_1, \ldots, t_i, \ldots, t_n) R f(t_1, \ldots, t'_i, \ldots, t_n)$ for any $n \geq 0$, $f \in \Sigma^{(n)}$, and $t_1, \ldots, t_n, t'_i \in T(\Sigma, X)$. 
Syntactic Quotient Algebras

Definition
Let \( \Sigma \) 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 \( J^R \) as follows:

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

Note
- \( (T(\Sigma, X)/R, J^R) \) is a \( \Sigma \)-algebra. Its carrier elements are sets of terms.
- The representatives are arbitrary: Let \( n \geq 0 \), \( f \in \Sigma^{(n)} \), and \( s_1, t_1, \ldots, s_n, t_n \in T(\Sigma, X) \). If \( s_i \sim t_i \), then \( f(s_1, \ldots, s_n) \sim f(t_1, \ldots, t_n) \). Hence, \( [f(s_1, \ldots, s_n)]_R = [f(t_1, \ldots, t_n)]_R \), as \( R \) is a congruence.

Example

Congruence classes of \( \approx_{\text{int}} \)

\[
\Sigma_{\text{int}} = \{ \text{zero}^{(0)}, \text{pred}^{(1)}, \text{succ}^{(1)} \}
\]

\[
\mathcal{E}_{\text{int}} = \{ \text{pred}(\text{succ}(x)) \approx x, \text{succ}(\text{pred}(x)) \approx x \}
\]

\[
[z\text{ero}]_{\approx_{\text{int}}} = \{ \text{zero}, \text{succ}(\text{pred}(\text{zero})), \text{pred}(\text{succ}(\text{zero})), \text{succ}(\text{succ}(\text{pred}(\text{pred}(\text{zero})))), \ldots \}
\]

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 \( \sigma \) and all \( s, t \in T(\Sigma, X) \).

Revised Definition for ADT Implementations

Definition
A datatype \( D = (M, J) \) implements ADT \( (\Sigma, \mathcal{E}) \) with constructors \( \Gamma \subseteq \Sigma \) if:
- \( (M, J) \) is a \( \Sigma \)-algebra (as before);
- All identities from \( \mathcal{E} \) are valid in \( M \) (as before);
- For all \( s, t \in T(\Gamma, \emptyset) \): \( s \approx_{\mathcal{E}} t \) iff \( J(s) = J(t) \) (new!)
Theorem
Let \((\Sigma, \mathcal{E})\) be an ADT with constructors \(\Gamma \subseteq \Sigma\). Then \(D = (T(\Sigma, \emptyset)/\approx, J^\approx)\) is an implementation of \((\Sigma, \mathcal{E})\).

Proof. Omitted

Example
\(\text{Nat as a constructor-based ADT (CADT)}\)
\[\text{CADT: } \Sigma = \{\text{zero, succ}\}, \mathcal{E} = \emptyset, \Gamma = \Sigma\]
Implementation: \((\mathbb{N}, J_1)\) with \(J_1(\text{zero})() = 0\) and \(J_1(\text{succ})(x) = x + 1\)
- \((\mathbb{N}, J_1)\) is \(\Sigma\)-algebra
- No identities to check
- Since \(\mathcal{E} = \emptyset, \approx\) is =.

\(\text{Dint}''\) is not an implementation of the natural numbers
\[\text{CADT: } \Sigma = \{\text{zero, succ}\}, \mathcal{E} = \emptyset, \Gamma = \Sigma\]
\[\{0, 1, 2, 3\}, J''\] with \(J''(\text{zero})() = 0, J''(\text{succ})(x) = (x + 1) \mod 4\) is not an implementation.
- \((\{0, 1, 2, 3\}, J'')\) is \(\Sigma\)-algebra
- No identities to check
- Since \(\mathcal{E} = \emptyset, \approx\) is =.

We have \(0 \neq \text{succ}^4(\text{zero})\) but \(J''(\text{zero}) = 0 = J''(\text{succ}^4(\text{zero}))\).

Example
\(\text{Alternative implementation of the natural numbers}\)
\[\text{CADT: } \Sigma = \{\text{zero, succ}\}, \mathcal{E} = \emptyset, \Gamma = \Sigma\]
Implementation: \((\{a\}^*, J'''\)) with \(J'''(\text{zero})() = \epsilon, J'''(\text{succ})(w) = aw\)
- \((\{a\}^*, J'''\)) is \(\Sigma\)-algebra
- No identities to check
- Since \(\mathcal{E} = \emptyset, \approx\) is =.

\(\text{If } s = t \text{ then } J'''(s) = J'''(t)\)
- Suppose \(s \neq t\). Then \(s = \text{succ}^n(t)\) with \(n > 0\). Hence, \(J'''(s) = J'''(t) + n \neq J'''(t)\).
Suppose \( \Gamma = \Sigma = \{ \text{zero}, \text{succ}, \text{pred} \} \),
\[ E = \{ \text{succ}(\text{pred}(x)) = x, \text{pred}(\text{succ}(x)) = x \} \]

**Question:** What is \( T(\Sigma, \emptyset)/\approx_E \)?

**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 \( t \in [\text{succ}^n(\text{zero})]_{\approx_E} \).

B \( t \in [\text{zero}]_{\approx_E} \).

C There exists \( n > 0 \) such that \( t \in [\text{pred}^n(\text{zero})]_{\approx_E} \).

**Proof (cont.)**

- **Induction Step for \( t = \text{succ}(t') \).** By the IH, one of the following holds for \( t' \):
  - A \( t' \approx_E \text{succ}^n(\text{zero}) \) for \( n > 0 \): then
    \[ \text{succ}(t') \approx_E \text{succ}^n(\text{zero}) = \text{succ}^{n+1}(\text{zero}). \]
    Since \( n + 1 > 0 \) we have case A.
  - B \( t' \approx_E \text{zero} \): then \( \text{succ}(t') \approx_E \text{succ}(\text{zero}) \). We have case A with \( n = 1 \).
  - C \( t' \approx_E \text{pred}^n(\text{zero}) \) for \( n > 0 \): then
    \[ \text{succ}(t') \approx_E \text{succ}(\text{pred}^n(\text{zero})). \]
    If \( n = 1 \) then \( \text{succ}(\text{pred}(\text{zero})) \approx_E \text{zero} \) so case B holds.
    If \( n > 1 \) then \( \text{succ}(\text{pred}(\text{pred}^{n-1}(\text{zero}))) \approx_E \text{pred}^{n-1}(\text{zero}) \), so case C holds.

**Proof (cont.)**

- **Induction step for \( \text{pred}(t) \) analogus.**
Equivalence Classes for Terms Representing Integers

Lemma
Suppose \( n > 0, m > 0 \). Then we have

- \( \text{zero} \not\approx_E \text{succ}^n(\text{zero}) \),
- \( \text{zero} \not\approx_E \text{pred}^n(\text{zero}) \),
- \( \text{succ}^n(\text{zero}) \not\approx_E \text{pred}^m(\text{zero}) \),
- \( \text{succ}^n(\text{zero}) \not\approx_E \text{succ}^m(\text{zero}) \) provided \( n \neq m \), and
- \( \text{pred}^n(\text{zero}) \not\approx_E \text{pred}^m(\text{zero}) \) provided \( n \neq m \).

If follows that

\[
\{\text{succ}^n(\text{zero})| n > 0\} \cup \{\text{zero}\} \cup \{\text{pred}^n(\text{zero})| n > 0\}
\]

is a set of representatives for \( T(\Sigma, \emptyset)/\approx_E \).

Example

Integers as a CADT

CADT: \( \Gamma = \Sigma = \{\text{zero}, \text{succ}, \text{pred}\}, \)
\( E = \{\text{succ} (\text{pred}(x)) = x, \text{pred} (\text{succ}(x)) = x\} \)

Implementation: \((Z, J)\) with
\( J(z) = 0, J(\text{succ}(x)) = x + 1, J(\text{pred}(x)) = x - 1 \)

- \((Z, J)\) is a \( \Sigma \)-algebra
- All identities are valid (as seen before)
- An easy term induction shows for all \( t \in T(\Sigma, \emptyset) \) that
  \- if \( t \approx_E \text{zero} \) then \( J(t) = 0 \),
  \- if \( t \approx_E \text{succ}^n(\text{zero}) \) then \( J(t) = n \), and
  \- if \( t \approx_E \text{pred}^n(\text{zero}) \) then \( J(t) = -n \).

Hence, if \( s \approx_E t \) then \( J(s) = J(t) \).
Conversely, if \( s \not\approx_E t \) then \( J(s) \neq J(t) \) because \( J \) maps different representatives to different integers.

Summary

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