Abstract Data Types
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 $(\Sigma, \mathcal{E})$ where
  - $\Sigma$ is a signature,
  - $\mathcal{E} \subseteq T(\Sigma, X) \times T(\Sigma, X)$ is a set of $\Sigma$-identities.
An ADT for natural numbers

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

\[ E_{nat} = \emptyset \]
Examples

An ADT for natural numbers

\[ \Sigma_{nat} = \{ \text{zero}^{(0)}, \text{succ}^{(1)} \} \]
\[ \mathcal{E}_{nat} = \emptyset \]

An ADT for integers

\[ \Sigma_{int} = \{ \text{zero}^{(0)}, \text{pred}^{(1)}, \text{succ}^{(1)} \} \]
\[ \mathcal{E}_{int} = \{ \text{pred(succ(x))} \approx x, \]
\[ \quad \text{succ(pred(x))} \approx x \} \]
Datatypes are \( \Sigma \)-Algebras

**Definition**

An identity \( s \approx t \) is **valid** in a \( \Sigma \)-algebra \( A = (A, \mathcal{J}) \) iff

\[
\mathcal{J}_\alpha(s) = \mathcal{J}_\alpha(t)
\]

for all variable assignments \( \alpha : X \rightarrow A \).
Datatypes are $\Sigma$-Algebras

**Definition**

- An identity $s \approx t$ is **valid** in a $\Sigma$-algebra $\mathcal{A} = (A, \mathcal{J})$ iff $\mathcal{J}_\alpha(s) = \mathcal{J}_\alpha(t)$ for all variable assignments $\alpha : X \rightarrow A$.
- A **datatype** is a $\Sigma$-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.)*
Implementations of the ADT for Naturals

Implementation 1

\( \mathcal{Dnat}' = (\mathbb{N}, \mathcal{J}') \), \( \mathcal{J}'(\text{zero}) = 0 \), \( \mathcal{J}'(\text{succ})(x) = x + 1 \) (note that \( x \) is meta-notation!).

- No identities, so all are valid.
- The function \( \mathcal{J}' \) is bijective.
Implementations of the ADT for Naturals

Implementation 1

\[ \mathcal{D}_{\text{nat'}} = (\mathbb{N}, J'), J'(\text{zero}) = 0, J'(\text{succ})(x) = x + 1 \]
(note that \( x \) is meta-notation!).

- No identities, so all are valid.
- The function \( J' \) is bijective.

Implementation 2

\[ \mathcal{D}_{\text{nat''}} = (\{0, 1, 2, 3\}, J''), J''(\text{zero}) = 0, J''(\text{succ})(x) = (x + 1) \mod 4. \]

- No identities, so all are valid.
- The function \( J'' \) is not injective (but surjective).

\[ J''(\text{zero}) = 0 = J''(\text{succ(\text{succ(\text{succ(\text{succ(\text{zero})})})})}) \]
Implementation 1

\[ \mathcal{D}_{\text{Int}}' = (\mathbb{Z}, \mathcal{J}'), \quad \mathcal{J}'(\text{zero})() = 0 \]
\[ \mathcal{J}'(\text{succ})(x) = x + 1 \]
\[ \mathcal{J}'(\text{pred})(x) = x - 1 \]

For arbitrary \( \alpha : \{x\} \rightarrow \mathbb{Z} \) we have
\[ \mathcal{J}'_{\alpha}(\text{pred}(\text{succ}(x))) = (\alpha(x) + 1) - 1 = \mathcal{J}'_{\alpha}(x) \]
\[ \mathcal{J}'_{\alpha}(\text{succ}(\text{pred}(x))) = (\alpha(x) - 1) + 1 = \mathcal{J}'_{\alpha}(x) \]

\( \mathcal{J}' \) is surjective but not injective. Consider
\[ \mathcal{J}'(\text{zero}) = 0 = \mathcal{J}'(\text{succ}(\text{pred}(\text{zero}))). \]
Implementations of the ADT for Integers (2)

Implementation 2

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

For arbitrary \( \alpha : \{x\} \rightarrow \mathbb{Z} \) we have
\[ J''(\text{pred}(\text{succ}(x)))) = J''(x) \]
\[ J''(\text{succ}(\text{pred}(x)))) = J''(x) \]

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

A Non-implementation

\[ Dint'''' = (\mathbb{N}, \mathcal{J}''') \]

\[ \mathcal{J}'''(\text{zero})() = 0 \]

\[ \mathcal{J}'''(\text{succ})(x) = x + 1 \]

\[ \mathcal{J}'''(\text{pred})(x) = \begin{cases} x - 1 & x > 0 \\ 0 & x = 0 \end{cases} \]

Not an implementation:
For \( \alpha : X \rightarrow \mathbb{N} \) with \( \alpha(x) = 0 \) we have

\[ \mathcal{J}_\alpha(\text{succ}(\text{pred}(x))) = 1 \neq 0 = \mathcal{J}_\alpha(x) \]
Fixing the Problems

- Want to rule out implementations such as $Dnat'$ and $Dint'$
- Definition of “implementation” is too weak
- Needed: restriction on function $\mathcal{J}$
  - $\mathcal{J}$ is not necessarily injective (see $Dint'$)
  - Idea: $\mathcal{J}$ must be injective on the equivalence classes induced by the identities of an ADT.
  In other words: $\mathcal{J}$ 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. $t_i \mathrel{R} t'_i$ implies $f(t_1, \ldots, t_i, \ldots, t_n) \mathrel{R} f(t_1, \ldots, t'_i, \ldots, t_n)$ for any $n \geq 0$, $f \in \Sigma^{(n)}$, and $t_1, \ldots, t_n, t'_i \in T(\Sigma, X)$. 
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 $s_1 R t_1, \ldots, s_n R t_n$ then $f(s_1, \ldots, s_n) R f(t_1, \ldots, t_n)$. Hence, $[f(s_1, \ldots, s_n)]_R = [f(t_1, \ldots, t_n)]_R$, as $R$ 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 \(s \approx t\);
- \(\approx_{\mathcal{E}}\) is closed under substitutions, i.e. \(s \approx t\) implies \(\sigma(s) \approx \sigma(t)\) for any substitution \(\sigma\) and all \(s, t \in T(\Sigma, X)\).
Congruence classes of $\approx_{E_{int}}$

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

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

\[ [\text{zero}]_{\approx_{E_{int}}} = \{ \text{zero}, \]
\[ \text{succ}(\text{pred}(\text{zero})), \]
\[ \text{pred}(\text{succ}(\text{zero})), \]
\[ \text{succ}(\text{succ}(\text{pred}(\text{pred}(\text{zero}))))), \ldots \]
Revised Definition for ADT Implementations

Definition

A datatype $D = (M, J)$ implements ADT $(\Sigma, E)$ with constructors $\Gamma \subseteq \Sigma$ if

- $(M, J)$ is a $\Sigma$-algebra (as before)
- All identities from $E$ are valid in $M$ (as before)
- For all $s, t \in T(\Gamma, \emptyset)$: $s \approx_E t$ iff $J(s) = J(t)$ (new!)
Theorem

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

*Proof.* Omitted
Nat as a constructor-based ADT (CADT)

CADT: $\Sigma = \{\text{zero, succ}\}, \mathcal{E} = \{\}, \Gamma = \Sigma$

Implementation: $(\mathbb{N}, \mathcal{J}_1)$ with $\mathcal{J}_1(\text{zero})() = 0$ and $\mathcal{J}_1(\text{succ})(x) = x + 1$

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

\(Dint''\) is not an implementation of the natural numbers

\[\text{CADT: } \Sigma = \{\text{zero, succ}\}, \mathcal{E} = \{\}, \Gamma = \Sigma\]
\[\{0, 1, 2, 3\}, \mathcal{J}''\) with \(\mathcal{J}''(\text{zero})() = 0,\]
\(\mathcal{J}''(\text{succ})(x) = (x + 1) \mod 4\) is not an implementation.

- \(\{0, 1, 2, 3\}, \mathcal{J}''\) is \(\Sigma\)-algebra
- No identities to check
- Since \(\mathcal{E} = \emptyset\), \(\approx_{\mathcal{E}}\) is =.
  We have \(\text{zero} \neq \text{succ}^4(\text{zero})\) but
  \(\mathcal{J}''(\text{zero}) = 0 = \mathcal{J}''(\text{succ}^4(\text{zero})).\)
Example

Alternative implementation of the natural numbers

CADT: $\Sigma = \{\text{zero}, \text{succ}\}, \mathcal{E} = \{\}, \Gamma = \Sigma$

Implementation: $(\{a\}^*, \mathcal{J}'''')$ with $\mathcal{J}''''(\text{zero})(\epsilon) = \epsilon$,
$\mathcal{J}''''(\text{succ})(w) = aw$

- $(\{a\}^*, \mathcal{J}'''')$ is $\Sigma$-algebra
- No identities to check
- Since $\mathcal{E} = \emptyset$, $\approx_\mathcal{E}$ is $=.$

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

**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 $t \in [\text{pred}^n(\text{zero})]\approx_E$. 
The proof is by term induction over $t$.

- Base case: $t = \text{zero}$. Then $B$ holds.
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 
\[
\text{succ}(t') \approx E \text{succ}(\text{succ}^{(n)}(\text{zero})) = \text{succ}^{(n+1)}(\text{zero}).
\]
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 
\[
\text{succ}(\text{pred}(\text{pred}^{(n-1)}(\text{zero})))) \approx E \text{pred}^{(n-1)}(\text{zero}), \text{so case C holds.} \]
Proof (cont.)

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

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

\begin{itemize}
\item zero \( \not\approx_\varepsilon \) \( \text{succ}^n(\text{zero}) \),
\item zero \( \not\approx_\varepsilon \) \( \text{pred}^n(\text{zero}) \),
\item \( \text{succ}^n(\text{zero}) \not\approx_\varepsilon \text{pred}^m(\text{zero}) \),
\item \( \text{succ}^n(\text{zero}) \not\approx_\varepsilon \text{succ}^m(\text{zero}) \) provided \( n \neq m \), and
\item \( \text{pred}^n(\text{zero}) \not\approx_\varepsilon \text{pred}^m(\text{zero}) \) provided \( n \neq m \).
\end{itemize}

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 \( T(\Sigma, \emptyset)/\approx_\varepsilon \).
Integers as a CADT

CADT: $\Gamma = \Sigma = \{\text{zero}, \text{succ}, \text{pred}\}$,
$\mathcal{E} = \{\text{succ}(\text{pred}(x)) = x, \text{pred}(\text{succ}(x)) = x\}$
Implementation: $(\mathbb{Z}, \mathcal{J})$ with
$\mathcal{J}(\text{z}) = 0$, $\mathcal{J}(\text{succ})(x) = x + 1$, $\mathcal{J}(\text{pred})(x) = x - 1$
- $(\mathbb{Z}, \mathcal{J})$ 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 $t \approx_{\mathcal{E}} \text{zero}$ then $\mathcal{J}(t) = 0$,
  - if $t \approx_{\mathcal{E}} \text{succ}^n(\text{zero})$ then $\mathcal{J}(t) = n$, and
  - if $t \approx_{\mathcal{E}} \text{pred}^n(\text{zero})$ then $\mathcal{J}(t) = -n$.
Hence, if $s \approx_{\mathcal{E}} t$ then $\mathcal{J}(s) = \mathcal{J}(t)$.
Conversely, if $s \not\approx_{\mathcal{E}} t$ then $\mathcal{J}(s) \neq \mathcal{J}(t)$ because $\mathcal{J}$ maps different representatives to different integers.
Summary

Slogan

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