Principles of Programming Languages
Lecture 2: Algebraic Data Types. Functions. Induction. Proofs.

Andrei Arusoaie[1]

[1]Department of Computer Science

October 5, 2021

Outline

ADTs

- ADT: Algebraic Data Type
- ADT = data types that are formed using other types
- Enumerated type example:

```
Inductive Season :=
  | Winter
  | Spring
  | Summer
  | Fall.
```

- This is a type with only four possible values
- The values are grouped into several classes (variants)
- Each variant has its own *constructor*

ADTs

- ADT: Algebraic Data Type
- ADT = data types that are formed using other types
- Enumerated type example:

```
Inductive Season :=
| Winter
| Spring
| Summer
| Fall.
```

- This is a type with only four possible values
- The values are grouped into several classes (variants)
- Each variant has its own *constructor*

ADTs

- ADT: Algebraic Data Type
- ADT = data types that are formed using other types
- Enumerated type example:

```
Inductive Season :=
| Winter
| Spring
| Summer
| Fall.
```

- This is a type with only four possible values
- The values are grouped into several classes (variants)
- Each variant has its own *constructor*

Inference rules

- Inference rules = *rule schemes*

$$\frac{I_1 \quad \cdots \quad I_n}{C}\ name$$

- Example: natural numbers

$$\frac{\cdot}{0 \in \mathbb{N}}\ zero \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ succ$$

- Inference rules can be used to define infinite sets
- We will use them to define the syntax and the semantics of programming languages

Inference rules

- Inference rules = *rule schemes*

$$\frac{I_1 \quad \cdots \quad I_n}{C}\ \textit{name}$$

- Example: natural numbers

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

- Inference rules can be used to define infinite sets
- We will use them to define the syntax and the semantics of programming languages

Inference rules

- Inference rules = *rule schemes*

$$\frac{I_1 \quad \cdots \quad I_n}{C}\ \textit{name}$$

- Example: natural numbers

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

- Inference rules can be used to define infinite sets
- We will use them to define the syntax and the semantics of programming languages

Inference rules

- Inference rules = *rule schemes*

$$\frac{I_1 \quad \cdots \quad I_n}{C}\ \textit{name}$$

- Example: natural numbers

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

- Inference rules can be used to define infinite sets
- We will use them to define the syntax and the semantics of programming languages

Inference rules

- Inference rules = *rule schemes*

$$\frac{I_1 \quad \cdots \quad I_n}{C}\ \textit{name}$$

- Example: natural numbers

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

- Inference rules can be used to define infinite sets
- We will use them to define the syntax and the semantics of programming languages

A type for naturals

- The syntax of natural numbers:

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

- In Coq:

```
Inductive Nat := O : Nat
               | S : Nat -> Nat.
```

- The first constructor takes no arguments
- The second constructor has arguments
- There is an infinite number of elements of this type

A type for naturals

- The syntax of natural numbers:

$$\frac{\cdot}{0 \in \mathbb{N}}\ \mathit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \mathit{succ}$$

- In Coq:

```
Inductive Nat := O : Nat
               | S : Nat -> Nat.
```

- The first constructor takes no arguments
- The second constructor has arguments
- There is an infinite number of elements of this type

A type for naturals

- The syntax of natural numbers:

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

- In Coq:

```
Inductive Nat := O : Nat
               | S : Nat -> Nat.
```

- The first constructor takes no arguments
- The second constructor has arguments
- There is an infinite number of elements of this type

Derivations

- Definition:

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

- Derivation example for $S\ (S\ O) \in \mathbb{N}$:

$$\dfrac{\dfrac{\dfrac{\cdot}{O \in \mathbb{N}}\ \textit{zero}}{S\ O \in \mathbb{N}}\ \textit{succ}}{S\ (S\ O) \in \mathbb{N}}\ \textit{succ}$$

- In Coq:

```
Check S (S O).
S (S O)
: Nat
```

Derivations

- Definition:

$$\frac{\cdot}{0 \in \mathbb{N}}\ zero \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ succ$$

- Derivation example for $S\ (S\ O) \in \mathbb{N}$:

$$\cfrac{\cfrac{\cfrac{\cdot}{O \in \mathbb{N}}\ zero}{S\ O \in \mathbb{N}}\ succ}{S\ (S\ O) \in \mathbb{N}}\ succ$$

- In Coq:

```
Check S (S O).
S (S O)
: Nat
```

Functions over naturals

- Recall syntax:

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

- Addition:

$$0 + m = m \qquad S\ n + m = S\ (n + m)$$

- In Coq:

```
Fixpoint plus (n m : Nat) : nat :=
match n with
    | 0 => m
    | S n' => S (plus n' m)
end.
```

Functions over naturals

- Recall syntax:

$$\frac{\cdot}{0 \in \mathbb{N}}\ \mathit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \mathit{succ}$$

- Addition:

$$0 + m = m \qquad\qquad S\ n + m = S\ (n + m)$$

- In Coq:

```
Fixpoint plus (n m : Nat) : nat :=
match n with
    | 0 => m
    | S n' => S (plus n' m)
end.
```

Evaluation

```
Eval compute in (plus O O).
= O
: Nat
Eval compute in (plus O (S O)).
= S O
: Nat
Eval compute in (plus (S O) O).
= S O
: Nat
Eval compute in (plus (S O) (S O)).
= S (S O)
: Nat
Eval compute in (plus (S (S O)) (S (S O))).
= S (S (S (S O)))
: Nat
```

Proof by simplification (demo)

```
Lemma plus_O_m_is_m :
  forall m, plus O m = m.
Proof.
  (* first, we move m from the quantifier
     in the goal to a context of current
     assumptions using the 'intro' tactic *)
  intros m.
  (* the current goal is now provable
     using the definition of plus: we
     apply the 'simpl' tactic for that*)
  simpl.
  (* the current goal is just reflexivity *)
  reflexivity.
 (* Qed checks if the proof is correct *)
Qed.
```

Proof by rewriting (demo)

```
Theorem plus_eq:
  forall m n, m = n -> plus O m = plus O n.
Proof.
  intros.
  rewrite <- H.
  reflexivity.
Qed.
```

Proof by case analysis, inversion (demo)

```
Theorem plus_c_a:
  forall k, plus k (S O) <> O.
Proof.
  intros.
  destruct k as [| k'] eqn:E.
  Show 2.
  - simpl. unfold not. intro H. inversion H.
  - simpl. unfold not. intro H. inversion H.
Qed.
```

Induction principles

- Recall the inductive definition for $\mathbb{N}$:

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

- What is a proof by induction in this case?
- Suppose that we want to prove $P(n)$ where $n \in \mathbb{N}$ and P is a property
- $P(n)$ holds if:
  - $P(0)$ holds - this corresponds to *zero*
  - $P(n)$ implies $P(n+1)$ - which corresponds to *succ*

Induction principles

- Recall the inductive definition for $\mathbb{N}$:

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

- What is a proof by induction in this case?
- Suppose that we want to prove $P(n)$ where $n \in \mathbb{N}$ and P is a property
- $P(n)$ holds if:
  - $P(0)$ holds - this corresponds to *zero*
  - $P(n)$ implies $P(n+1)$ - which corresponds to *succ*

Induction principles

- Recall the inductive definition for $\mathbb{N}$:

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

- What is a proof by induction in this case?
- Suppose that we want to prove $P(n)$ where $n \in \mathbb{N}$ and P is a property
- $P(n)$ holds if:
  - $P(0)$ holds - this corresponds to *zero*
  - $P(n)$ implies $P(n+1)$ - which corresponds to *succ*

Induction principles

- For naturals defined as below $\mathbb{N}$:

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

  there is a corresponding **induction principle**:

$$\forall P.\ \big(P(0) \wedge (\forall n.P(n) \to P(S\ n))\big) \to \forall n.P(n)$$

- Proofs by induction are based on induction principles

Induction principles in Coq

- In Coq induction principles are generated automatically
- Recall the Coq definition of naturals:

```
Inductive Nat := O : Nat
               | S : Nat -> Nat.
```

- When the definition is processed, Coq generates (among others) the `Nat_ind` induction principle (demo)
- `Check Nat_ind.`

```
Nat_ind
     : forall P : Nat -> Prop,
       P O ->
       (forall n : Nat, P n -> P (S n)) ->
       forall n : Nat, P n
```

Proofs by induction: example 1

- Lemma: $P = \forall n. n + 0 = n$
- Can we prove this by applying the definitions?

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

$0 + m = m$ (base) $\qquad$ $S\ n + m = S\ (n + m)$ (ind)

- Induction principle:
  $\forall P. \left(P(0) \wedge (\forall n. P(n) \rightarrow P(S\ n))\right) \rightarrow \forall n. P(n)$
- Proof by induction
  1. case $P(0) : 0 + 0 = 0$, by (base)
  2. case $n = S\ n'$:
     - Inductive hypothesis (IH): $P(n) : n + 0 = n$
     - Prove $P(S\ n) : S\ n + 0 =$ (by ind) $S\ (n + 0) =$ (by IH) $= S\ n$

Proofs by induction: example 1

- Lemma: $P = \forall n. n + 0 = n$
- Can we prove this by applying the definitions?

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

$0 + m = m$ (base) $\qquad S\ n + m = S\ (n + m)$ (ind)

- Induction principle:
  $\forall P. \left(P(0) \land (\forall n. P(n) \rightarrow P(S\ n))\right) \rightarrow \forall n. P(n)$
- Proof by induction
  1. case $P(0) : 0 + 0 = 0$, by (base)
  2. case $n = S\ n'$:
     - Inductive hypothesis (IH): $P(n) : n + 0 = n$
     - Prove $P(S\ n) : S\ n + 0 =$ (by ind) $S\ (n + 0) =$ (by IH) $= S\ n$

Proofs by induction: example 1

- Lemma: $P = \forall n . n + 0 = n$
- Can we prove this by applying the definitions?

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

$$0 + m = m \text{ (base)} \qquad\qquad S\ n + m = S\ (n + m) \text{ (ind)}$$

- Induction principle:
  $\forall P. \left(P(0) \wedge (\forall n . P(n) \rightarrow P(S\ n))\right) \rightarrow \forall n . P(n)$
- Proof by induction
  1. case $P(0) : 0 + 0 = 0$, by (base)
  2. case $n = S\ n'$:
     - Inductive hypothesis (IH): $P(n) : n + 0 = n$
     - Prove $P(S\ n) : S\ n + 0 =$ (by ind) $S\ (n + 0) =$ (by IH) $= S\ n$

Proofs by induction: example 1

- Lemma: $P = \forall n.n + 0 = n$
- Can we prove this by applying the definitions?

$$\frac{\cdot}{0 \in \mathbb{N}}\ zero \qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ succ$$

$$0 + m = m \text{ (base)} \qquad S\ n + m = S\ (n + m) \text{ (ind)}$$

- Induction principle:
  $\forall P. \big(P(0) \wedge (\forall n.P(n) \rightarrow P(S\ n))\big) \rightarrow \forall n.P(n)$
- Proof by induction
  1. case $P(0) : 0 + 0 = 0$, by (base)
  2. case $n = S\ n'$:
     - Inductive hypothesis (IH): $P(n) : n + 0 = n$
     - Prove $P(S\ n) : S\ n + 0 =$ (by ind) $S\ (n + 0) =$ (by IH) $= S\ n$

Proofs by induction: example 1

- Lemma: $P = \forall n. n + 0 = n$
- Can we prove this by applying the definitions?

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

$$0 + m = m \text{ (base)} \qquad\qquad S\ n + m = S\ (n + m) \text{ (ind)}$$

- Induction principle:
  $\forall P. \big(P(0) \wedge (\forall n. P(n) \rightarrow P(S\ n))\big) \rightarrow \forall n. P(n)$
- Proof by induction
  1. case $P(0) : 0 + 0 = 0$, by (base)
  2. case $n = S\ n'$:
     - Inductive hypothesis (IH): $P(n) : n + 0 = n$
     - Prove $P(S\ n)$: $S\ n + 0 =$ (by ind) $S\ (n + 0) =$ (by IH) $= S\ n$

Proofs by induction: example 1

- Lemma: $P = \forall n. n + 0 = n$
- Can we prove this by applying the definitions?

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad\qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

$$0 + m = m \text{ (base)} \qquad\qquad S\ n + m = S\ (n + m) \text{ (ind)}$$

- Induction principle:
  $\forall P. \left(P(0) \wedge (\forall n. P(n) \rightarrow P(S\ n))\right) \rightarrow \forall n. P(n)$
- Proof by induction
  1. case $P(0) : 0 + 0 = 0$, by (base)
  2. case $n = S\ n'$:
     - Inductive hypothesis (IH): $P(n) : n + 0 = n$
     - Prove $P(S\ n) : S\ n + 0 =$ (by ind) $S\ (n + 0) =$ (by IH) $= S\ n$

Proofs by induction: example 1

- Lemma: $P = \forall n. n + 0 = n$
- Can we prove this by applying the definitions?

$$\frac{\cdot}{0 \in \mathbb{N}}\ \mathit{zero} \qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \mathit{succ}$$

$$0 + m = m \text{ (base)} \qquad S\ n + m = S\ (n + m) \text{ (ind)}$$

- Induction principle:
  $\forall P. \big(P(0) \wedge (\forall n. P(n) \rightarrow P(S\ n))\big) \rightarrow \forall n. P(n)$
- Proof by induction
  1. case $P(0) : 0 + 0 = 0$, by (base)
  2. case $n = S\ n'$:
     - Inductive hypothesis (IH): $P(n) : n + 0 = n$
     - Prove $P(S\ n) : S\ n + 0 =$ (by ind) $S\ (n + 0) =$ (by IH) $= S\ n$

Proofs by induction: example 1

- Lemma: $P = \forall n. n + 0 = n$
- Can we prove this by applying the definitions?

$$\frac{\cdot}{0 \in \mathbb{N}}\ \textit{zero} \qquad \frac{n \in \mathbb{N}}{S\ n \in \mathbb{N}}\ \textit{succ}$$

$$0 + m = m \text{ (base)} \qquad S\ n + m = S\ (n + m) \text{ (ind)}$$

- Induction principle:
  $\forall P. \big(P(0) \wedge (\forall n. P(n) \to P(S\ n))\big) \to \forall n. P(n)$
- Proof by induction
  1. case $P(0) : 0 + 0 = 0$, by (base)
  2. case $n = S\ n'$:
     - Inductive hypothesis (IH): $P(n) : n + 0 = n$
     - Prove $P(S\ n) : S\ n + 0 =$ (by ind) $S\ (n + 0) =$ (by IH) = $S\ n$

Proof by induction: example 1

```
Lemma plus_n_O_is_n :
  forall n,
    plus n O = n.
Proof.
  (* Exercise: why this does not work by simplification? *)
  (* Induction by n using Nat_ind principle:
     note that two goals are generated. *)
  induction n.
  - (* solve the first goal by simplification *)
    simpl.
    reflexivity.
  - (* simplifies just a part of the goal *)
    simpl.
    (* apply the inductive hypothesis by
       conveniently rewriting the expression *)
    rewrite IHn.
    reflexivity.
Qed.
```

Proof by induction: example 2

```
Lemma plus_comm :
  forall n m, plus n m = plus m n.
(* DEMO *)
```

Demo: lists in Coq

- Defining lists in Coq
- Induction principle
- Functions: append, reverse
- Proofs: append nil left/right, associative append, involutive reverse

Where do we need ADTs?

Example: ASTs

- AST = Abstract Syntax Trees
- An AST is a *tree*-like representation of the structure of a program
- "Abstract" = not every detail occuring in the text of the program is present in the AST representation
- Compilers use ASTs as the main data structure

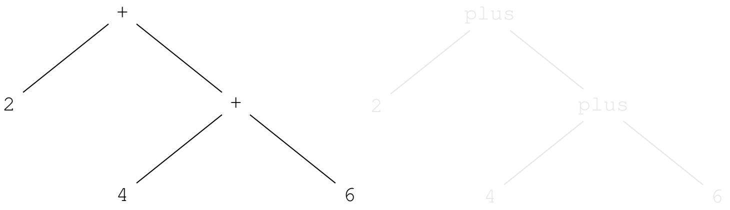
Where do we need ADTs?

Example: ASTs

- AST = Abstract Syntax Trees
- An AST is a *tree*-like representation of the structure of a program
- "Abstract" = not every detail occuring in the text of the program is present in the AST representation
- Compilers use ASTs as the main data structure

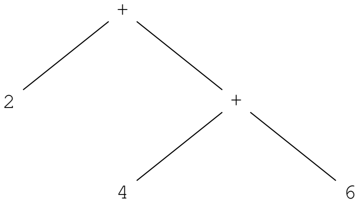
Example: arithmetic expressions

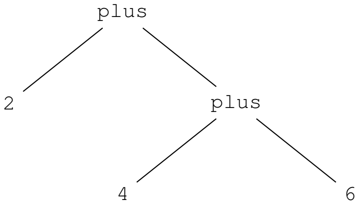
AST for `2 + (4 + 6)`:



Example: arithmetic expressions

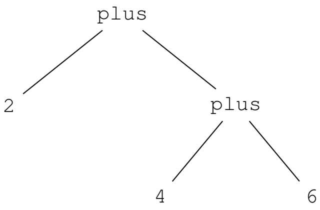
AST for `2 + (4 + 6)`:





Example: arithmetic expressions

AST for `2 + (4 + 6)`:




```
Inductive Exp :=
| number : nat -> Exp
| plus : Exp -> Exp -> Exp.

Coercion number : nat >-> Exp.

Check (plus 2 (plus 4 6)).
plus 2 (plus 4 6)
     : Exp
```

Conclusions

- Intensively used concepts:
  1. inductive definitions
  2. induction principles
  3. functions
  4. proofs
- Bibliography:

  1. Chaper Inductive Definitions in Practical Foundations of Programming Languages, Robert Harper
     https://www.cs.cmu.edu/~rwh/pfpl/2nded.pdf
  2. Chapter Proof by induction in Software Foundations - Volume 1, Benjamin C. Pierce, Arthur Azevedo de Amorim, Chris Casinghino, Marco Gaboardi, Michael Greenberg, Cătălin Hriţcu, Vilhelm Sjöberg, Andrew Tolmach, Brent Yorgey
     https://softwarefoundations.cis.upenn.edu/lf-current/Induction.html

Conclusions

- Intensively used concepts:
  1. inductive definitions
  2. induction principles
  3. functions
  4. proofs
- Bibliography:

  1. Chaper Inductive Definitions in Practical Foundations of Programming Languages, Robert Harper
     https://www.cs.cmu.edu/~rwh/pfpl/2nded.pdf
  2. Chapter Proof by induction in Software Foundations - Volume 1, Benjamin C. Pierce, Arthur Azevedo de Amorim, Chris Casinghino, Marco Gaboardi, Michael Greenberg, Cătălin Hriţcu, Vilhelm Sjöberg, Andrew Tolmach, Brent Yorgey
     https://softwarefoundations.cis.upenn.edu/lf-current/Induction.html