Did I Get it Right?
Part 4: Induction for Datatypes

Speaker: David Walker
COS 326
Princeton University

http://~cos326/notes/reasoning-data.php
Equational Reasoning: Some Key Ideas

What is the fundamental definition of expression equality \((e_1 == e_2)\)?

- two expressions are equal if:
  - they evaluate to equal values, or
  - they both raise the same exception
  - they both fail to terminate
- note: we won’t ask you to do proofs about expressions that don't terminate, use I/O or mutable data structures

What are some consequences of this definition?

- expression equality is reflexive, symmetric and transitive
- if \(e_1 \rightarrow e_2\) then \(e_1 == e_2\)
- if \(e_1 == e_2\) then \(e[e_1/x] == e[e_2/x]\). (substitution of equals for equals)

How do we prove things about recursive functions?

- we use proofs by induction
- to reason about recursive calls on smaller data, we assume the property we are trying to prove (i.e., we use the induction hypothesis)
More General Template for Inductive Datatypes

\[
\text{type } t = \text{C1 of } t_1 \mid \text{C2 of } t_2 \mid \ldots \mid \text{Cn of } t_n
\]

Types \( t_1, t_2 \ldots t_n \), may contain 1 or more occurrences of \( t \) within them.

Examples:

- \( \text{type mylist = MyNil} \mid \text{MyCons of int * mylist} \)
- \( \text{type 'a tree = Leaf} \mid \text{Node of 'a * 'a tree * 'a tree} \)
More General Template for Inductive Datatypes

**Theorem:** For all x : t, property(x).

**Proof:** By induction on structure of values x with type t.
More General Template for Inductive Datatypes

Theorem: For all $x : t$, property(x).

Proof: By induction on structure of values $x$ with type $t$.

Case: $x == C1 v$:

... use IH on components of $v$ that have type $t$ ...

Case: $x == C2 v$:

... use IH on components of $v$ that have type $t$ ...

Case: $x == Cn v$:

... use IH on components of $v$ that have type $t$ ...
A PROOF ABOUT TREES
type 'a tree = Leaf | Node of 'a * 'a tree * 'a tree

let rec tm f t =
  match t with
  | Leaf -> Leaf
  | Node (x, l, r) -> Node (f x, tm f l, tm f r)

let (<> f g =
  fun x -> f (g x)
Another example

```ocaml
type 'a tree = Leaf | Node of 'a * 'a tree * 'a tree

let rec tm f t =
  match t with
    | Leaf -> Leaf
    | Node (x, l, r) -> Node (f x, tm f l, tm f r)

let (<>) f g =
  fun x -> f (g x)
```

**Theorem:**
For all (total) functions \( f : b \rightarrow c \),
For all (total) functions \( g : a \rightarrow b \),
For all trees \( t : a \text{ tree} \),
\( \text{tm } f \ (\text{tm } g \ t) = \text{tm } (f <> g) \ t \)
Theorem:
For all (total) functions \( f : b \rightarrow c \),
For all (total) functions \( g : a \rightarrow b \),
For all trees \( t : a \) tree,
\( \text{tm} \ f \ (\text{tm} \ g \ t) = \text{tm} \ (f <> g) \ t \)

```
let rec tm f t =
  match t with
  | Leaf -> Leaf
  | Node (x, l, r) -> Node (f x, tm f l, tm f r)

let (<>) f g =
  fun x -> f (g x)
```

To begin, let’s *pick an arbitrary total function \( f \) and total function \( g \).*
We’ll prove the theorem without assuming any particular properties of \( f \) or \( g \) (other than the fact that the types match up). So, for the \( f \) and \( g \) we picked, we’ll prove:

**Theorem:**
For all trees \( t : a \) tree,
\( \text{tm} \ f \ (\text{tm} \ g \ t) = \text{tm} \ (f <> g) \ t \)
Theorem:
For all trees \( t \) : a tree,
\( \text{tm } f \ (\text{tm } g \ t) == \text{tm } (f <> g) \ t \)

```ocaml
let rec tm f t = 
  match t with 
  | Leaf -> Leaf 
  | Node (x, l, r) -> Node (f x, tm f l, tm f r) 

let (<> f g = 
  fun x -> f (g x)
```
Another example

Theorem:
For all trees $t : \text{ a tree,}$
$tm\ f\ (tm\ g\ t) == tm\ (f\ <>\ g)\ t$

Case: $t = \text{ Leaf}$

No inductive hypothesis to use.
(Leaf doesn’t contain any smaller components with type tree.)

Proof:
$tm\ f\ (tm\ g\ \text{Leaf})$

let rec tm f t =
  match t with
  | Leaf -> Leaf
  | Node (x, l, r) -> Node (f x, tm f l, tm f r)

let (<>) f g =
  fun x -> f (g x)
Another example

Theorem:
For all trees \( t : \text{a tree} \),
\[
\text{tm } f \ (\text{tm } g \ t) = \text{tm } (f <> g) \ t
\]

Case: \( t = \text{Leaf} \)

No inductive hypothesis to use.
(Leaf doesn’t contain any smaller components with type tree.)

Proof:
\[
\text{tm } f \ (\text{tm } g \ \text{Leaf}) \\
== \text{tm } f \ \text{Leaf} \quad \text{(eval } \text{tm } g \ \text{Leaf}) \\
== \text{Leaf} \quad \text{(eval } \text{tm } f \ \text{Leaf}) \\
== \text{tm } (f <> g) \ \text{Leaf} \quad \text{(reverse eval)}
\]

let rec tm f t =
match t with
| Leaf -> Leaf
| Node (x, l, r) -> Node (f x, tm f l, tm f r)

let (<>) f g =
fun x -> f (g x)
Another example

Theorem:
For all trees t : a tree,
tm f (tm g t) == tm (f <> g) t

Case: t = Node(v, l, r)

IH1: tm f (tm g l) == tm (f <> g) l
IH2: tm f (tm g r) == tm (f <> g) r

let rec tm f t =
match t with
  | Leaf -> Leaf
  | Node (x, l, r) -> Node (f x, tm f l, tm f r)

let (<>) f g =
  fun x -> f (g x)
Another example

Theorem:
For all trees $t : a$ tree,
$t \mapsto f \cdot tm \cdot g \cdot t = tm \cdot (f \cdot <> \cdot g) \cdot t$

Case: $t = \text{Node}(v, l, r)$

IH1: $tm \cdot f \cdot (tm \cdot g \cdot l) == tm \cdot (f \cdot <> \cdot g) \cdot l$

IH2: $tm \cdot f \cdot (tm \cdot g \cdot r) == tm \cdot (f \cdot <> \cdot g) \cdot r$

Proof:
$tm \cdot f \cdot (tm \cdot g \cdot (\text{Node} \: (v, l, r)))$

$== tm \cdot (f \cdot <> \cdot g) \cdot (\text{Node} \: (v, l, r))$

let rec $tm \; f \; t =$
  match $t$ with
  | $\text{Leaf}$ -> $\text{Leaf}$
  | $\text{Node} \: (x, l, r)$ -> $\text{Node} \: (f \; x, \; tm \; f \; l, \; tm \; f \; r)$

let $(<>)$ $f \; g =$
  fun $x$ -> $f \; (g \; x)$
Another example

Theorem:
For all trees \( t : \) a tree,
\[
\text{tm } f \ (\text{tm } g \ t) = \text{tm } (f \ <> \ g) \ t
\]

Case: \( t = \text{Node}(v, l, r) \)

IH1: \( \text{tm } f \ (\text{tm } g \ l) = \text{tm } (f \ <> \ g) \ l \)
IH2: \( \text{tm } f \ (\text{tm } g \ r) = \text{tm } (f \ <> \ g) \ r \)

Proof:
\[
\text{tm } f \ (\text{tm } g \ (\text{Node}(v, l, r)))
\]
\[
= \text{tm } f \ (\text{Node}(g \ v, \text{tm } g \ l, \text{tm } g \ r))
\]
\[
= \text{tm } (f \ <> \ g) \ (\text{Node}(v, l, r))
\]
Another example

Theorem:
For all trees $t : \text{a tree},$
$\text{tm } f (\text{tm } g \ t) == \text{tm } (f <> g) \ t$

Case: $t = \text{Node}(v, l, r)$

IH1: $\text{tm } f (\text{tm } g \ l) == \text{tm } (f <> g) \ l$
IH2: $\text{tm } f (\text{tm } g \ r) == \text{tm } (f <> g) \ r$

Proof:
$\text{tm } f (\text{tm } g \ (\text{Node } (v, l, r)))$
$== \text{tm } f (\text{Node } (g \ v, \text{tm } g \ l, \text{tm } g \ r))$  \hspace{2cm} (eval inner tm)

$\text{Node } ((f <> g) \ v, \text{tm } (f <> g) \ l, \text{tm } (f <> g) \ r)$
$== \text{tm } (f <> g) \ (\text{Node } (v, l, r))$  \hspace{2cm} (eval reverse)

let rec tm f t =
match t with
  | Leaf -> Leaf
  | Node (x, l, r) -> Node (f x, tm f l, tm f r)

let (<>) f g =
fun x -> f (g x)
Another example

Theorem:
For all trees $t : \text{a tree}$,
$\text{tm} \ f \ (\text{tm} \ g \ t) = \text{tm} \ (f <> g) \ t$

Case: $t = \text{Node}(v, l, r)$

IH1: $\text{tm} \ f \ (\text{tm} \ g \ l) = \text{tm} \ (f <> g) \ l$
IH2: $\text{tm} \ f \ (\text{tm} \ g \ r) = \text{tm} \ (f <> g) \ r$

Proof:

$\text{tm} \ f \ (\text{tm} \ g \ (\text{Node} \ (v, l, r)))$
$= \text{tm} \ f \ (\text{Node} \ (g \ v, \text{tm} \ g \ l, \text{tm} \ g \ r))$
$= \text{Node} \ (f \ (g \ v), \text{tm} \ f \ (\text{tm} \ g \ l), \text{tm} \ f \ (\text{tm} \ g \ r))$

(eval inner tm)
(eval – since g, tm are total)

$\text{Node} \ ((f <> g) \ v, \text{tm} \ (f <> g) \ l, \text{tm} \ (f <> g) \ r)$
$= \text{tm} \ (f <> g) \ (\text{Node} \ (v, l, r))$

(eval reverse)

let rec tm f t =
match t with
| Leaf -> Leaf
| Node (x, l, r) -> Node (f x, tm f l, tm f r)

let (<> f g =
  fun x -> f (g x)
Another example

orem:
For all trees \( t : \) a tree,
\[ \text{tm} \ f \ (\text{tm} \ g \ t) \ == \ \text{tm} \ (f <> g) \ t \]

Case: \( t = \text{Node}(v, l, r) \)

IH1: \( \text{tm} \ f \ (\text{tm} \ g \ l) \ == \ \text{tm} \ (f <> g) \ l \)
IH2: \( \text{tm} \ f \ (\text{tm} \ g \ r) \ == \ \text{tm} \ (f <> g) \ r \)

Proof:
\[
\begin{align*}
\text{tm} \ f \ (\text{tm} \ g \ (\text{Node} \ (v, l, r))) &
\quad == \quad \text{tm} \ f \ (\text{Node} \ (g \ v, \text{tm} \ g \ l, \text{tm} \ g \ r)) \\
&
\quad == \quad \text{Node} \ ((f <> g) \ v, \text{tm} \ (f <> g) \ l, \text{tm} \ (f <> g) \ r) \\
&
\quad == \quad tm \ (f <> g) \ (\text{Node} \ (v, l, r))
\end{align*}
\]

let rec tm f t =
    match t with
    | Leaf -> Leaf
    | Node (x, l, r) -> Node (f x, tm f l, tm f r)

let (<> f g =
    fun x -> f (g x)
Another example

Theorem:
For all trees t : a tree,
\( \text{tm } f \ (\text{tm } g \ t) \equiv \text{tm } (f <> g) \ t \)

Case: \( t = \text{Node}(v, l, r) \)

IH1: \( \text{tm } f \ (\text{tm } g \ l) \equiv \text{tm } (f <> g) \ l \)
IH2: \( \text{tm } f \ (\text{tm } g \ r) \equiv \text{tm } (f <> g) \ r \)

Proof:
\[
\begin{align*}
\text{tm } f \ (\text{tm } g \ (\text{Node} \ (v, l, r))) \\
\equiv \text{tm } f \ (\text{Node} \ (g \ v, \text{tm } g \ l, \text{tm } g \ r)) \\
\equiv \text{Node} \ ((f <> g) \ v, \text{tm } f \ (\text{tm } g \ l), \text{tm } f \ (\text{tm } g \ r)) \\
\equiv \text{Node} \ ((f <> g) \ v, \text{tm } (f <> g) \ l, \text{tm } f \ (\text{tm } g \ r)) \\
\equiv \text{Node} \ ((f <> g) \ v, \text{tm } (f <> g) \ l, \text{tm } (f <> g) \ r) \\
\equiv \text{tm } (f <> g) \ (\text{Node} \ (v, l, r))
\end{align*}
\]

(let rec tm f t =
  match t with
  | Leaf -> Leaf
  | Node (x, l, r) -> Node (f x, tm f l, tm f r)

let (<> ) f g =
  fun x -> f (g x)
Theorem:
For all trees $t : a$ tree,
$tm\ f\ (tm\ g\ t) == tm\ (f\ <>\ g)\ t$

Case: $t = Node(v, l, r)$

IH1: $tm\ f\ (tm\ g\ l) == tm\ (f\ <>\ g)\ l$
IH2: $tm\ f\ (tm\ g\ r) == tm\ (f\ <>\ g)\ r$

Proof:
$tm\ f\ (tm\ g\ (Node\ (v, l, r)))$
$== tm\ f\ (Node\ (g\ v, tm\ g\ l, tm\ g\ r))\quad (eval\ inner\ tm)$
$== Node\ (f\ (g\ v),\ tm\ f\ (tm\ g\ l),\ tm\ f\ (tm\ g\ r))\quad (eval\ –\ since\ g,\ tm\ are\ total)$
$== Node\ ((f\ <>\ g)\ v,\ tm\ f\ (tm\ g\ l),\ tm\ f\ (tm\ g\ r))\quad (eval\ reverse)$
$== Node\ ((f\ <>\ g)\ v,\ tm\ (f\ <>\ g)\ l,\ tm\ (f\ <>\ g)\ r)\quad (IH1)$
$== Node\ ((f\ <>\ g)\ v,\ tm\ (f\ <>\ g)\ l,\ tm\ (f\ <>\ g)\ r)\quad (IH2)$
$== tm\ (f\ <>\ g)\ (Node\ (v, l, r))\quad (eval\ reverse)$
Proof Template for Trees

**Theorem:** For all \( x : \text{a tree} \), \( \text{property}(x) \).

**Proof:** By induction on the structure of trees \( x \).

**Case:** \( x == \text{Leaf} \):

... no use of inductive hypothesis (this is the smallest tree) ...

**Case:** \( x == \text{Node} (v, \text{left}, \text{right}) \):

IH1: \( \text{property}(`\text{left}) \)
IH2: \( \text{property}(\text{right}) \)

... use IH1 and IH 2 in your proof ...
**Summary of Template for Inductive Datatypes**

**Theorem:** For all \( x : t \), \( \text{property}(x) \).

**Proof:** By induction on structure of values \( x \) with type \( t \).

**Case:** \( x == \text{C1} \, v \):

... use \( \text{IH} \) on components of \( v \) that have type \( t \) ...

**Case:** \( x == \text{C2} \, v \):

... use \( \text{IH} \) on components of \( v \) that have type \( t \) ...

**Case:** \( x == \text{Cn} \, v \):

... use \( \text{IH} \) on components of \( v \) that have type \( t \) ...

use patterns that divide up the cases

Take inspiration from the structure of the program
Exercise

type 'a tree = Leaf of 'a | Node of 'a tree * 'a tree

let rec flip (t: 'a tree) =
match t with
| Leaf _ -> t
| Leaf _ -> t
| Node (a,b) -> Node (flip b, flip a)

Theorem: for all t: 'a tree, flip(flip t) = t.

Theorem: for all t: 'a tree, flip(flip (flip t)) = flip t.