Data types with Negation

Robert Atkey
University of Strathclyde
robert.atkey@strath.ac.uk

Workshop on Mathematically Structured Functional Programming
2nd April 2022

```
data Nat : Set where
  zero : Nat
  succ : Nat → Nat
```

But what if we could use negation?

```
data Even : Nat → Set where
  zero     : Even zero
  succ-odd : ∀{n} → Odd n → Even (succ n)

data Odd : Nat → Set where
  succ-even : ∀{n} → Even n → Odd (succ n)
```

```
data Even : Nat → Set where
  zero : Even zero
  succ : ∀{n} → not (Even n) → Even (succ n)
```

Representing Parse Trees

A *Context-Free Grammar*:

$$
\begin{aligned}
S ::= & \ A \\
| & \ B
\end{aligned}
$$

Representing Parse Trees

A *Context-Free Grammar*:

$$\begin{aligned} S ::= & \ A \\ | & \ B \end{aligned}$$

Representation of the parse trees:

$$\begin{aligned} &\mathsf{data}\ \mathrm{S} : \mathrm{String} \times \mathrm{String} \to \mathrm{Set}\ \mathsf{where} \\ &\quad \mathsf{prod1} : \forall \{i\, o\} \to \mathrm{A}(i, o) \to \mathrm{S}(i, o) \\ &\quad \mathsf{prod2} : \forall \{i\, o\} \to \mathrm{B}(i, o) \to \mathrm{S}(i, o) \end{aligned}$$

Representing Parse Trees

A *Parsing Expression Grammar*:

$$S ::= A \;/\; B$$

"Parse S as an A, or *if that fails*, as a B"

Representing Parse Trees

A *Parsing Expression Grammar*:

$$S ::= A \;/\; B$$

"Parse S as an A, or *if that fails*, as a B"

$$\begin{aligned}
&\mathsf{data}\ \mathrm{S} : \mathrm{String} \times \mathrm{String} \to \mathrm{Set}\ \mathsf{where} \\
&\quad \mathsf{prod1} : \forall \{i\ o\} \to \mathrm{A}(i, o) \to \mathrm{S}(i, o) \\
&\quad \mathsf{prod2} : \forall \{i\ o\} \to \mathrm{not}\ (\exists o'.\ \mathrm{A}(i, o')) \to \mathrm{B}(i, o) \to \mathrm{S}(i, o)
\end{aligned}$$

```
data Liar : Set where
  liar : not Liar → Liar
```

```
data Liar : Set where
  liar : not Liar → Liar
```

?

Semantics of Inductive Types

1. Construct a functor $F : (I \to \text{Set}) \to (I \to \text{Set})$ from the definition
   - Essentially a sum-of-products construction

2. The interpretation of the data type is the initial F-algebra
   - $\Rightarrow$ unique "fold"
   - $\Rightarrow$ derive induction principles

$$\text{not } A \stackrel{def}{=} A \to \bot$$

$$\text{not } A \stackrel{def}{=} A \to \bot$$

?

$$\mathsf{not}\ A \stackrel{def}{=} A \to \bot$$

?

Not covariant: don't get a functor!

What do we mean when we say “not”?

1. “This leads to a contradiction”

$$\text{not } A \stackrel{def}{=} A \to \bot$$

What do we mean when we say “not”?

1. “This leads to a contradiction”

$$\text{not } A \stackrel{def}{=} A \to \bot$$

2. “I do not have any evidence to believe this”

What do we mean when we say “not”?

1. “This leads to a contradiction”

$$\text{not } A \stackrel{def}{=} A \rightarrow \bot$$

2. “I do not have any evidence to believe this”

$$\text{files}(X) \leftarrow \text{bird}(X), \text{not penguin}(X).$$
$$\text{bird}(\text{tweety}).$$

What do we mean when we say “not”?

1. “This leads to a contradiction”

$$\text{not } A \stackrel{def}{=} A \rightarrow \bot$$

2. “I do not have any evidence to believe this”

$$\text{files}(X) \leftarrow \text{bird}(X), \text{not penguin}(X).$$
$$\text{bird}(\text{tweety}).$$

Does “tweety” fly?

What do we mean when we say “not”?

1. “This leads to a contradiction”

$$\text{not } A \stackrel{def}{=} A \to \bot$$

2. “I do not have any evidence to believe this”

$$\text{files}(X) \leftarrow \text{bird}(X), \text{not penguin}(X).$$
$$\text{bird}(\text{tweety}).$$

Does “tweety” fly?

Classically (and intuitionistically): no.

What do we mean when we say “not”?

1. “This leads to a contradiction”

$$\text{not } A \stackrel{def}{=} A \to \bot$$

2. “I do not have any evidence to believe this”

$$\text{files}(X) \leftarrow \text{bird}(X), \text{not penguin}(X).$$
$$\text{bird}(\text{tweety}).$$

Does “tweety” fly?

Classically (and intuitionistically): no. Intuitively – yes?

What do we mean when we say “not”?

1. “This leads to a contradiction”

$$\text{not } A \stackrel{def}{=} A \to \bot$$

2. “I do not have any evidence to believe this”

$$\text{files}(X) \leftarrow \text{bird}(X), \text{not penguin}(X).$$
$$\text{bird}(\text{tweety}).$$

Does “tweety” fly?

Classically (and intuitionistically): no. Intuitively – yes?

Warning this kind of reasoning is *non monotonic*. If we later learn penguin(tweety), then we would have to retract our prior conclusion.

Constructive Falsity

A way to track evidence for and evidence *against* some data:

$$A = (A^+, A^-)$$

Constructive Falsity

A way to track evidence for and evidence *against* some data:

$$A = (A^+, A^-)$$

Entailment flows forwards positively and backwards negatively:

$$(A^+, A^-) \Rightarrow (B^+, B^-) \stackrel{def}{=} (A^+ \to B^+) \times (B^- \to A^-)$$

call this category Chu.

Constructive Falsity

A way to track evidence for and evidence *against* some data:

$$A = (A^+, A^-)$$

Entailment flows forwards positively and backwards negatively:

$$(A^+, A^-) \Rightarrow (B^+, B^-) \stackrel{def}{=} (A^+ \to B^+) \times (B^- \to A^-)$$

call this category Chu.

Aside: we could also add the condition $A^+ \times A^- \to \bot$, but will skip this for now.

Connectives

1. Conjunction

$$(A^+, A^-) \times (B^+, B^-) \stackrel{def}{=} (A^+ \times B^+, A^- + B^-)$$

2. Disjunction

$$(A^+, A^-) + (B^+, B^-) \stackrel{def}{=} (A^+ + B^+, A^- \times B^-)$$

3. Infinitary Conjunction

$$\Pi x{:}X.\, A[x] \stackrel{def}{=} (\Pi x{:}X.\, A^+[x], \Sigma x{:}X.\, A^-[x])$$

4. Negation

$$\text{not}\,(A^+, A^-) \stackrel{def}{=} (A^-, A^+)$$

5. Sets

$$[X] = (X, X \to \bot)$$

Initial Algebras in Chu

If we have

$$F : (I \to \mathrm{Chu}) \to (I \to \mathrm{Chu})$$

constructed from only $\times, +, \Pi, \Sigma, [-]$ then it can be separated:

$$F^+ : (I \to \mathrm{Set}) \to (I \to \mathrm{Set}) \qquad F^- : (I \to \mathrm{Set}) \to (I \to \mathrm{Set})$$

and Initial algebras in Chu can be constructed from those in Set:

$$\mu F = (\mu F^+, \nu F^-)$$

$$
\begin{aligned}
&\mathsf{data}\ \mathrm{Path} : \mathrm{Node} \times \mathrm{Node} \to \mathrm{Set}\ \mathsf{where} \\
&\quad \mathsf{stop} : \forall x \to \mathrm{Path}(x, x) \\
&\quad \mathsf{step} : \forall x\, y\, z \to \mathrm{Path}(x, y) \to \mathrm{Step}(y, z) \to \mathrm{Path}(x, z)
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{data}\ \mathrm{Path} : \mathrm{Node} \times \mathrm{Node} \longrightarrow \mathrm{Set}\ \mathsf{where} \\
&\quad \mathsf{stop} : \forall x \longrightarrow \mathrm{Path}(x, x) \\
&\quad \mathsf{step} : \forall x\, y\, z \longrightarrow \mathrm{Path}(x, y) \longrightarrow \mathrm{Step}(y, z) \longrightarrow \mathrm{Path}(x, z)
\end{aligned}
$$

$$
F(X)(x, z) = [x = z] + (\Sigma y.\ X(x, y) \times [\mathrm{Step}(y, z)])
$$

$$\mathsf{data}\ \mathrm{Path} : \mathrm{Node} \times \mathrm{Node} \longrightarrow \mathrm{Set}\ \mathsf{where}$$
$$\mathsf{stop} : \forall x \longrightarrow \mathrm{Path}(x, x)$$
$$\mathsf{step} : \forall x\, y\, z \longrightarrow \mathrm{Path}(x, y) \longrightarrow \mathrm{Step}(y, z) \longrightarrow \mathrm{Path}(x, z)$$

$$F(X)(x, z) = [x = z] + (\Sigma y.\ X(x, y) \times [\mathrm{Step}(y, z)])$$

$$F^{+}X^{+}(x, z) = (x = z) + (\Sigma y.\ X^{+}(x, y) \times \mathrm{Step}(y, z))$$
$$F^{-}X^{-}(x, z) = \neg(x = z) \times (\Pi y.\ X^{-}(x, y) + \neg\mathrm{Step}(y, z))$$

$$\mathsf{data}\ \mathrm{Path} : \mathrm{Node} \times \mathrm{Node} \longrightarrow \mathrm{Set}\ \mathsf{where}$$
$$\mathsf{stop} : \forall x \longrightarrow \mathrm{Path}(x, x)$$
$$\mathsf{step} : \forall x\, y\, z \longrightarrow \mathrm{Path}(x, y) \longrightarrow \mathrm{Step}(y, z) \longrightarrow \mathrm{Path}(x, z)$$

$$F(X)(x, z) = [x = z] + (\Sigma y.\ X(x, y) \times [\mathrm{Step}(y, z)])$$

$$F^{+}X^{+}(x, z) = (x = z) + (\Sigma y.\ X^{+}(x, y) \times \mathrm{Step}(y, z))$$
$$F^{-}X^{-}(x, z) = \neg(x = z) \times (\Pi y.\ X^{-}(x, y) + \neg\mathrm{Step}(y, z))$$

$$\mathrm{Path} = (\mu F^{+}, \nu F^{-})$$

What do you want?

What about data types with negation?

Information!

Define another category Chu^i with the same objects but:

$$(A^+, A^-) \overset{i}{\Rightarrow} (B^+, B^-) = (A^+ \to B^+, A^- \to B^-)$$

Information morphisms – parallel transformation of positive and negative information.

Negation is *covariant* in Chu^i.

Information!

Define another category Chu^i with the same objects but:

$$(A^+, A^-) \overset{i}{\Rightarrow} (B^+, B^-) = (A^+ \to B^+, A^- \to B^-)$$

Information morphisms – parallel transformation of positive and negative information.

Negation is *covariant* in Chu^i.

Initial F-Algebras in Chu^i:

$$\mu^i F = \mu(X^+, X^-).(F^+(X^+, X^-), F^-(X^+, X^-))$$

Information!

Define another category Chu^i with the same objects but:

$$(A^+, A^-) \overset{i}{\Rightarrow} (B^+, B^-) = (A^+ \to B^+, A^- \to B^-)$$

Information morphisms – parallel transformation of positive and negative information.

Negation is *covariant* in Chu^i.

Initial F-Algebras in Chu^i:

$$\mu^i F = \mu(X^+, X^-).(F^+(X^+, X^-), F^-(X^+, X^-))$$

You won't get it! Works for all of our functors, but gives the "wrong" answer.

Idea:

Refine the positive meaning of a data type with respect to information about non-provability.

Idea:

Refine the positive meaning of a data type with respect to information about non-provability.

The *reduct* of a logic program from **Stable Model Semantics**

(Gelfond and Lifschitz, 1989).

Idea:

Refine the positive meaning of a data type with respect to information about non-provability.

The *reduct* of a logic program from **Stable Model Semantics**

(Gelfond and Lifschitz, 1989).

Given $F : (I \to \mathrm{Chu}) \to (I \to \mathrm{Chu})$

Idea:

Refine the positive meaning of a data type with respect to information about non-provability.

The *reduct* of a logic program from **Stable Model Semantics**

(Gelfond and Lifschitz, 1989).

Given $F : (I \to \mathrm{Chu}) \to (I \to \mathrm{Chu})$ and $Y : I \to \mathrm{Chu}$, define

Idea:

Refine the positive meaning of a data type with respect to information about non-provability.

The *reduct* of a logic program from **Stable Model Semantics**

(Gelfond and Lifschitz, 1989).

Given $F : (I \to \mathrm{Chu}) \to (I \to \mathrm{Chu})$ and $Y : I \to \mathrm{Chu}$, define

$$F/Y(X^+, X^-) = \lambda i.\, (F^+(X^+, Y^-)\, i, F^-(Y^+, X^-)\, i)$$

Y represents "a stage of knowledge".

Idea:

Refine the positive meaning of a data type with respect to information about non-provability.

The *reduct* of a logic program from **Stable Model Semantics**

(Gelfond and Lifschitz, 1989).

Given $F : (I \to \mathrm{Chu}) \to (I \to \mathrm{Chu})$ and $Y : I \to \mathrm{Chu}$, define

$$F/Y(X^+, X^-) = \lambda i.\, (F^+(X^+, Y^-)\, i, F^-(Y^+, X^-)\, i)$$

Y represents "a stage of knowledge".

F/Y is separable, so we can get $\mu(F/Y) : I \to \mathrm{Chu}$.

Idea:

Refine the positive meaning of a data type with respect to information about non-provability.

The *reduct* of a logic program from **Stable Model Semantics**

(Gelfond and Lifschitz, 1989).

Given $F : (I \to \mathrm{Chu}) \to (I \to \mathrm{Chu})$ and $Y : I \to \mathrm{Chu}$, define

$$F/Y(X^+, X^-) = \lambda i.\, (F^+(X^+, Y^-)\, i, F^-(Y^+, X^-)\, i)$$

Y represents "a stage of knowledge".

F/Y is separable, so we can get $\mu(F/Y) : I \to \mathrm{Chu}$.

Moreover, get a functor $\mu(F/-) : \mathrm{Chu}^I \to \mathrm{Chu}^I$.

A Sematics of Data types with Negation

Given $F : (I \to \mathrm{Chu}) \to (I \to \mathrm{Chu})$ for a data type D, define

$$\begin{aligned} D &= \mu^i Y.\, \mu(F/Y) \\ &= \mu(Y^+, Y^-).(\mu X^+.\, F^+(X^+, Y^-), \nu X^-.\, F^-(Y^+, X^-)) \end{aligned}$$

A Sematics of Data types with Negation

Given $F : (I \to \mathrm{Chu}) \to (I \to \mathrm{Chu})$ for a data type D, define

$$\begin{aligned} D &= \mu^i Y.\, \mu(F/Y) \\ &= \mu(Y^+, Y^-).(\mu X^+.\, F^+(X^+, Y^-), \nu X^-.\, F^-(Y^+, X^-)) \end{aligned}$$

If we replace Set by Bool, and add the $\mathrm{Chu}(\perp)$ constraint, then this coincides with the 3-valued stable model semantics (Przymusinski, 1990).

Example: Liar

```
data Liar : Set where
  liar : not Liar → Liar
```

Example: Liar

```
data Liar : Set where
  liar : not Liar → Liar
```

$$FX = \text{not} X$$

Example: Liar

```
data Liar : Set where
  liar : not Liar → Liar
```

$$FX = \text{not} X$$

$$F^{+}(X^{+}, X^{-}) = X^{-} \qquad F^{-}(X^{+}, X^{-}) = X^{+}$$

Example: Liar

$$\begin{aligned}&\mathsf{data}\ \mathsf{Liar} : \mathsf{Set}\ \mathsf{where}\\ &\quad \mathsf{liar} : \mathsf{not}\ \mathsf{Liar} \to \mathsf{Liar}\end{aligned}$$

$$FX = \mathsf{not} X$$

$$F^+(X^+, X^-) = X^- \qquad F^-(X^+, X^-) = X^+$$

$$\begin{aligned}\mathsf{Liar} &= \mu^i(Y^+, Y^-).(\mu X^+.\, F^+(X^+, Y^-), \nu X^-.\, F^-(Y^+, X^-))\\ &= \mu^i(Y^+, Y^-).(Y^-, Y^+)\\ &\cong (\bot, \bot)\end{aligned}$$

Example: Identity

```
data NoBaseCase : Set where
  rec : NoBaseCase → NoBaseCase
```

Example: Identity

```
data NoBaseCase : Set where
  rec : NoBaseCase → NoBaseCase
```

$$FX = X$$

Example: Identity

```
data NoBaseCase : Set where
  rec : NoBaseCase → NoBaseCase
```

$$FX = X$$

$$\begin{aligned} \text{NoBaseCase} &= \mu(Y^{+}, Y^{-}).(\mu X^{+}.\, X^{+}, \nu X^{-}.\, X^{-}) \\ &\cong (\bot, \top) \end{aligned}$$

Example: Identity

```
data NoBaseCase : Set where
  rec : NoBaseCase → NoBaseCase
```

$$FX = X$$

$$\begin{aligned} \text{NoBaseCase} &= \mu(Y^{+}, Y^{-}).(\mu X^{+}.\, X^{+}, \nu X^{-}.\, X^{-}) \\ &\cong (\bot, \top) \end{aligned}$$

No proofs, one (extensionally) refutation.

Example: Even

```
data Even : Nat → Set where
  zero : Even zero
  succ : ∀{n} → not (Even n) → Even (succ n)
```

Example: Even

$$
\begin{aligned}
&\mathsf{data}\ \mathrm{Even} : \mathrm{Nat} \to \mathrm{Set}\ \mathsf{where} \\
&\quad \mathrm{zero} : \mathrm{Even}\ \mathrm{zero} \\
&\quad \mathrm{succ} : \forall\{n\} \to \mathrm{not}\ (\mathrm{Even}\ n) \to \mathrm{Even}\ (\mathrm{succ}\ n)
\end{aligned}
$$

$$
F X = \lambda\{\mathrm{zero} \mapsto \top ; \mathrm{succ}\ n \mapsto \mathrm{not}\ (X\, n)\}
$$

Example: Even

$$
\begin{aligned}
&\mathsf{data}\ \mathsf{Even} : \mathsf{Nat} \longrightarrow \mathsf{Set}\ \mathsf{where} \\
&\quad \mathsf{zero} : \mathsf{Even}\ \mathsf{zero} \\
&\quad \mathsf{succ} : \forall\{n\} \longrightarrow \mathsf{not}\ (\mathsf{Even}\ n) \longrightarrow \mathsf{Even}\ (\mathsf{succ}\ n)
\end{aligned}
$$

$$
FX = \lambda\{\mathsf{zero} \mapsto \top; \mathsf{succ}\ n \mapsto \mathsf{not}\ (X\, n)\}
$$

$$
\begin{aligned}
F^{+}(X^{+}, X^{-}) &= \lambda\{\mathsf{zero} \mapsto \top; \mathsf{succ}\ n \mapsto X^{-}\ n\} \\
F^{-}(X^{+}, X^{-}) &= \lambda\{\mathsf{zero} \mapsto \bot; \mathsf{succ}\ n \mapsto X^{+}\ n\}
\end{aligned}
$$

Example: Even

$$
\begin{aligned}
&\mathsf{data}\ \mathrm{Even} : \mathrm{Nat} \to \mathrm{Set}\ \mathsf{where} \\
&\quad \mathsf{zero} : \mathrm{Even}\ \mathsf{zero} \\
&\quad \mathsf{succ} : \forall\{n\} \to \mathrm{not}\ (\mathrm{Even}\ n) \to \mathrm{Even}\ (\mathsf{succ}\ n)
\end{aligned}
$$

$$
FX = \lambda\{\mathsf{zero} \mapsto \top; \mathsf{succ}\ n \mapsto \mathrm{not}\ (X\, n)\}
$$

$$
\begin{aligned}
F^{+}(X^{+}, X^{-}) &= \lambda\{\mathsf{zero} \mapsto \top; \mathsf{succ}\ n \mapsto X^{-}\ n\} \\
F^{-}(X^{+}, X^{-}) &= \lambda\{\mathsf{zero} \mapsto \bot; \mathsf{succ}\ n \mapsto X^{+}\ n\}
\end{aligned}
$$

$$
\mathrm{Even} = \mu(Y^{+}, Y^{-}).\ (\lambda\{\mathsf{zero} \mapsto \top; \mathsf{succ}\ n \mapsto Y^{-}\ n\}, \lambda\{\mathsf{zero} \mapsto \bot; \mathsf{succ}\ n \mapsto Y^{+}\ n\})
$$

which is $\cong$ to the mutually defined Even/Odd definition.

Conclusion

- Constructed a plausible semantics of data types with negation
- Based on 3-valued stable model semantics of logic programming
- Potentially useful for modelling backtracking processes

Conclusion

- Constructed a plausible semantics of data types with negation
- Based on 3-valued stable model semantics of logic programming
- Potentially useful for modelling backtracking processes

Related work:

- Constructive Falsity: “anithesis translation” Shulman (2018);
- Weak negation / negation as failure: Clark (1978), Gelfond and Lifschitz (1988), Przymusinski (1989)
- Bilattices: Ginsberg (1986), Fitting (2020)

Conclusion

- Constructed a plausible semantics of data types with negation
- Based on 3-valued stable model semantics of logic programming
- Potentially useful for modelling backtracking processes

Related work:

- Constructive Falsity: “anithesis translation” Shulman (2018);
- Weak negation / negation as failure: Clark (1978), Gelfond and Lifschitz (1988), Przymusinski (1989)
- Bilattices: Ginsberg (1986), Fitting (2020)

- TODO: Implementation (underway)
- TODO: Reasoning Principles