Expressing and Implementing the Computational Content Implicit in Smullyan's Account of Boolean Valuations

Matthew Fluet

November 25, 2003

Introduction

- Computational Aspects of Logic
  - "pure" approach
  - "computational" approach
- Algorithm Development

Outline

- Introduction
- Preliminaries
- Four Approaches
  - The Mathematician
  - The Programmer
  - The Formalist
  - The Computer Scientist
- Conclusion

Formulas

ABS: form

```
form() ==
  rec(form.$pvar:$v:ℕ +
           $pnot:$U:form +
           $pand:$U:form × $V:form +
           $por:$U:form × $V:form +
           $pimp:$U:form × $V:form)
```

The notation `$l:t` labels a term with a token parameter.

These tokens are used by abstract syntax tree tactics to label case analyses, invoke appropriate folding/unfolding of definitions, and to generate names for introduced variables.

Formulas

- definitions for injections into the `form()` type
- definition for case discrimination of the `form()` type
- tactics for induction (structural, size, and depth)
- decidability theorems for equality and membership

All generated automatically from a succinct description of the desired abstract syntax.

Valuations and Extensions

Now we consider, in addition to the formulas of propositional logic, a set $\{t, f\}$ of two *distinct* elements, t, f as *truth-values*. For any set S of formulas, by a *valuation* of S, we mean a function v from S into the set $\{t, f\}$ – i.e., a mapping which assigns to every element X of S one of the two values t, f.

If S_1 is a subset of S_2 and if v_1, v_2 are respective valuations of S_1, S_2, then we say that v_2 is an *extension* of v_1 if v_2, v_1 agree on the smaller set S_1.

ABS: ValuationExtension

```
ValuationExtension(S1;f1;S2;f2) ==  ∀X:S1. f1 X = f2 X ∈ 𝔹
```

STM: ValutionExtension_wf

```
∀S1:𝕌. ∀f1:S1 ⟶ 𝔹. ∀S2:𝕌. ∀f2:S2 ⟶ 𝔹.
  ((S1 ⊆r S2) ⇒ (ValuationExtension(S1;f1;S2;f2) ∈ ℙ))
```

Boolean Valuations

Now we wish to consider valuations of the set E of all formulas of propositional logic. We are not really interested in *all* valuations of E, but only in those which are "faithful" to the usual "truth-table" rules for the logical connectives. This idea we make precise in the following defintion.

Definition 1. A valuation v of E is called a *Boolean* valuation if for every X, Y in E, the following conditions hold:

B_1**:** The formula $\neg X$ receives the gvalue t if X receives the value f and f if X receives the value t.

B_2**:** The formula $X \wedge Y$ receives the value t if X, Y both receive the value t, otherwise $X \wedge Y$ receives the value f.

B_3**:** The formula $X \vee Y$ receives the value t if at least one of X, Y receives the value t, otherwise $X \vee Y$ receives the value f.

B_2**:** The formula $X \Rightarrow Y$ receives the value f if X, Y receive the respective values t, f, otherwise $X \Rightarrow Y$ receives the value t.

This concludes our definition of a Boolean valuation.

Boolean Valuations

ABS: PartialBooleanValuation

```
PartialBooleanValuation(S;f) ==
  ∀X:S.form_cases(X;v.True;Y.f X = ¬b(f Y) ∈ 𝔹;
                  Y,Z.f X = (f Y) ∧b (f Z) ∈ 𝔹;
                  Y,Z.f X = (f Y) ∨b (f Z) ∈ 𝔹;
                  Y,Z.f X = f Y ⇒b (f Z) ∈ 𝔹)
```

Boolean Valuations

ABS: DownClosed

```
DownClosedForm(S) ==  ∀X:S. (SubForm(X) ⊆r S)
```

STM: DownClosed_wf

```
∀S:{S:𝕌  S ⊆r form()}. (DownClosedForm(S) ∈ ℙ)
```

STM: PartialBooleanValuation_wf

```
∀S:{S:𝕌  S ⊆r form()}.
  (DownClosedForm(S) ⇒
   (∀f:S ⟶ 𝔹. (PartialBooleanValuation(S;f) ∈ ℙ)))
```

Existence and Uniqueness of Boolean Valuations

STM: ###

```
∀Z:form(). ∀v0:Var(SubForm(Z)) ⟶ 𝔹.∃!f:SubForm(Z) ⟶ 𝔹.
  ValuationExtension(Var(SubForm(Z));v0;SubForm(Z);f) ∧
  PartialBooleanValuation(SubForm(Z);f)
```

Criteria

- Elegance
- Extract
- Effort

Four Approaches

- The Mathematician – “pure” approach
- The Programmer – “computational” w/o extracts
- The Formalist – “compuatational” w/ extracts
- The Computer Scientist – “computational” w/ v.d.f.

The Mathematician

Consider a single formula X and an interpretation v_0 of X – or for that matter any assignment v_0 of truth values to a set of propositional variables which includes at least all variables of X (and possibly others). It is easily verified by induction on the degree of X that there exists one and only one way of assigning truth values to all *subformulas* of X such that the *atomic* subformulas of X (which are propositional variables) are assigned the same truth values as under v_0, and such that the truth value of each *compound* subformula Y of X is determined from the truth values of the immediate subformulas of Y by the truth-table rules $B_1 - B_4$.

The Mathematician

```
PRF: BySmullyan_UniqueExistence
# 3
 .....pand case.....
 1. Q : form() ⟶ ℙ
 2. ∀Z:{Z:form()  Q Z}. ∀v0:Var(SubForm(Z)) ⟶ 𝔹.
      {!f:SubForm(Z) ⟶ 𝔹
        ValuationExtension(Var(SubForm(Z));v0;SubForm(Z);f)
        ∧ PartialBooleanValuation(SubForm(Z);f)}
 3. U : {Z:form()  Q Z}
 4. V : {Z:form()  Q Z}
 5. v0 : Var(SubForm(pand(U;V))) ⟶ 𝔹
 ⊢ {!f:SubForm(pand(U;V)) ⟶ 𝔹
      ValuationExtension(Var(SubForm(pand(U;V)));v0;
                         SubForm(pand(U;V));f)
      ∧ PartialBooleanValuation(SubForm(pand(U;V));f)}
 BY ???
```

The Mathematician

PRF: BySmullyan_UniqueExistence

```
λZ.if form_equalF <Z, pand(U;V)> then (fU U) ∧_b (fV V)
   if IsSubFormF <Z, U> then fU Z
   if IsSubFormF <Z, V> then fV Z
   else tt
   fi
```

The Programmer

CODE: Value_ml

```
Value(v0;X)
==r form_cases(X;v.v0 pvar(v); A.¬_b Value(v0;A);
               A,B.Value(v0;A) ∧_b Value(v0;B);
               A,B.Value(v0;A) ∨_b Value(v0;B);
               A,B.Value(v0;A) ⇒_b Value(v0;B))
```

PRF: ByExplicit_UniqueExistence_view

```
#
 TERMOF{ByExplicit_UniqueExistence:o, \v:l}
 BY RepeatFor 1 (RW (HigherC AnyExtractC) 0) THEN Reduce 0·
 # 1
 λZ,v0,X.Value(v0;X)
```

The Formalist

```
STM: ByExtract_Uniqueness
∀Z:form(). ∀v0:Var(SubForm(Z)) ⟶ 𝔹.
  {y:𝔹
   ∀f:SubForm(Z) ⟶ 𝔹.
     ((ValuationExtension(Var(SubForm(Z));v0;SubForm(Z);f) ∧
       PartialBooleanValuation(SubForm(Z);f))
     ⇒ y = f Z ∈ 𝔹)}
```

The Formalist

PRF: ByExtract_Uniqueness_view

```
#
 TERMOF{ByExtract_Uniqueness:o, \v:l}
 BY RepeatFor 2 (RW (HigherC AnyExtractC) 0) THEN Reduce 0·
 # 1
 λZ.rec_ind(Z;f,x.form_cases(x;v.λv0.(v0 pvar(v));
                              X.λv0.(¬b(f X v0));
                              X,Y.λv0.((f X v0) ∧b (f Y v0));
                              X,Y.λv0.((f X v0) ∨b (f Y v0));
                              X,Y.λv0.(f X v0 ⇒b (f Y v0))))
```

The Computer Scientist

```
ABS: RFunctionValue
```

$$
\begin{array}{l}
\texttt{RFunctionValue(X;v0) ==} \\
\quad \texttt{\{f}\quad \texttt{A:SubForm(X)} \longrightarrow \texttt{\{form\_cases(A;v.v0 pvar(v);Y.}\neg_b\texttt{(f Y);} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{Y,Z.(f Y)} \wedge_b \texttt{(f Z);} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{Y,Z.(f Y)} \vee_b \texttt{(f Z);} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{Y,Z.f Y} \Rightarrow_b \texttt{(f Z)):}\mathbb{B}\texttt{\}\}}
\end{array}
$$

- show `RFunctionValue` is a type
- show `RFunctionValue` is inhabitted
- show any inhabitant of `RFunctionValue` is a Boolean valuation

The Computer Scientist

Complications

- no rule for inverting dependent set membership (singleton types)
- no rule for interpreting a very dependent function type as a "normal" function type
- symptomatic of the relative infancy of the very-dependent function type (e.g., lack of tactic support)

The Computer Scientist

PRF: RFunctionValue_Existence_view

```
#
 TERMOF{RFunctionValue_Existence:o, \v:l}
 BY RepeatFor 2 (RW (SweepUpC AnyExtractC) 0) THEN Reduce 0·
 # 1
 λZ,v0,A.(Y (λf.form_cases(A;v.v0 pvar(v);Y.¬b(f Y);
                            Y,Z.(f Y) ∧b (f Z);
                            Y,Z.(f Y) ∨b (f Z);
                            Y,Z.f Y ⇒b (f Z))))
```

Conclusion

- Strive for mutual satisfaction
  - mathematicians – more comfortable manipulating “pure” functions than algorithm descriptions
  - programmers – want to verify the correctness of an efficient evalution algorithm
  - formalists – wish to extract the evaluation algorithm from the appopriate existence proof