Evaluation Trees for Proposition Algebra

Alban Ponse

–

joined work with Jan A. Bergstra

section Theory of Computer Science

Informatics Institute, University of Amsterdam

https://staff.fnwi.uva.nl/a.ponse/

ERO'60 - September 9, 2015

1. Introduction

Short-circuit evaluation (SE) in imperative programming:

if (*not* `(j==0) && (i/j > 17)`) then (..) else (..)

Clearly, SE is sequential and `&&` (Logical and) is not commutative...

Questions:

Q1. For conditions as above: which are the logical laws that characterize SE?

Q2. As Q1, but restricting to atoms that evaluate to either `true` or `false` (either exclude atoms as `(i/j > 17)`, or **require** such evalutions)

Q3. As Q2, but involving constants T and F for `true` and `false`

An example that falsifies idempotency of `&&` (programmable in `Perl`):

1) For program variable `i`, atom `(i==`k`)` with $k \in \mathbb{Z}$ is a test, and

2) Boolean evaluation of assignment `(i=e)` yields `false` iff `e`'s value is 0.

Then, if `i` has initial value 2,

`(i=i+1) && (i==3)` evaluates to `true`, and

`((i=i+1) && (i=i+1)) && (i==3)` evaluates to `false`

Wrt. Q2 and Q3, some logical laws that are not valid: (equational)

- Idempotency, thus $x \mathbin{\&\&} x = x$ and $x \mathbin{||} x = x$, where $||$ represents "Logical or"
- Distributivity, e.g. $x \mathbin{\&\&} (y \mathbin{||} z) = (x \mathbin{\&\&} y) \mathbin{||} (x \mathbin{\&\&} z)$
- Absorption, e.g. $x \mathbin{\&\&} (x \mathbin{||} y) = x$

Towards a systematic answer of Q2 and Q3:

- Involve *Hoare's conditional* (1985), a ternary connective characterized by

$$P \triangleleft Q \triangleright R \approx \textit{if } Q \textit{ then } P \textit{ else } R$$

With the conditional, one can define negation and the (binary) propositional connectives that prescribe SE:

$$\begin{aligned} \neg x &= F \triangleleft x \triangleright T \\ x \mathbin{\&\&} y &= y \triangleleft x \triangleright F \\ x \mathbin{||} y &= T \triangleleft x \triangleright y \end{aligned}$$

Fact: basic equational axioms for the conditional imply $\neg\neg x = x$ (DNS), associativity of the propositional connectives, and De Morgan's laws.

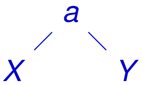
2. Evaluation trees

$CProp(A)$, Conditional Propositions with atoms in A:

$$P ::= a \mid T \mid F \mid P \triangleleft P \triangleright P \quad (a \in A).$$

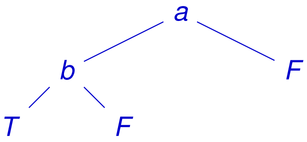
$\mathcal{T}_A$, *Evaluation trees over* A, provide a simple semantics for $CProp(A)$:

$$X ::= T \mid F \mid X \trianglelefteq a \trianglerighteq X \quad (a \in A).$$

Pictorial representation: 

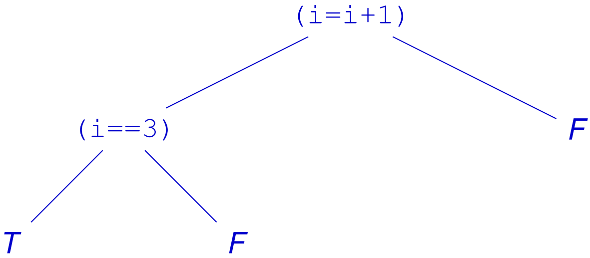
for $X \trianglelefteq a \trianglerighteq Y$

Thus: binary trees with leaves in $\{T, F\}$ and internal nodes in A, e.g.



OR $(T \trianglelefteq b \trianglerighteq F) \trianglelefteq a \trianglerighteq F$

Idea: For `(i=i+1) && (i==3)`, thus for `(i==3)` $\triangleleft$ `(i=i+1)` $\triangleright$ F, an evaluation is a complete path in the evaluation tree



where

- the evaluation starts in the root node `(i=i+1)`, and continues in the **left** branch if `(i=i+1)` evaluates to `true`, and otherwise in the **right** branch
- evaluation in the internal node `(i==3)` proceeds likewise
- **leaves** represent the final evaluation value

Leaf replacement in $X \in \mathcal{T}_A$, notation

$$X\,[T \mapsto Y, F \mapsto Z]$$

is defined by

$$T\,[T \mapsto Y, F \mapsto Z] = Y$$

$$F\,[T \mapsto Y, F \mapsto Z] = Z$$

$$(X_1 \trianglelefteq a \trianglerighteq X_2)\,[T \mapsto Y, F \mapsto Z] = \begin{array}{c} a \\ \swarrow \qquad \searrow \\ X_1\,[T \mapsto Y, F \mapsto Z] \qquad X_2\,[T \mapsto Y, F \mapsto Z] \end{array}$$

The *short-circuit interpretation function* $se : CProp(A) \to \mathcal{T}_A$ is defined by

$$
\begin{aligned}
se(T) &= T \\
se(F) &= F \\
se(a) &= T \trianglelefteq a \trianglerighteq F \\
se(P \triangleleft Q \triangleright R) &= se(Q)\,[T \mapsto se(P), F \mapsto se(R)]
\end{aligned}
$$

Example:

$$se(F \triangleleft a \triangleright T) = (T \trianglelefteq a \trianglerighteq F)\,[T \mapsto F, F \mapsto T] = F \trianglelefteq a \trianglerighteq T = \begin{array}{ccc} & a & \\ \diagup & & \diagdown \\ F & & T \end{array}$$

Thus, $se(F \triangleleft a \triangleright T)$ models the evaluation of $\neg a$, and we can involve negation by

$$se(\neg P) = se(P)\,[T \mapsto F, F \mapsto T]$$

CP, a set of axioms for $.. \triangleleft .. \triangleright ..$ (*Proposition algebra* [BP10]):

$$x \triangleleft T \triangleright y = x$$
$$x \triangleleft F \triangleright y = y$$
$$T \triangleleft x \triangleright F = x$$
$$x \triangleleft (y \triangleleft z \triangleright u) \triangleright v = (x \triangleleft y \triangleright v) \triangleleft z \triangleright (x \triangleleft u \triangleright v)$$

Example: $CP \vdash F \triangleleft (F \triangleleft x \triangleright T) \triangleright T = (F \triangleleft F \triangleright T) \triangleleft x \triangleright (F \triangleleft T \triangleright T)$
$= T \triangleleft x \triangleright F$
$= x$

and thus with $\neg x = F \triangleleft x \triangleright T$ we find DNS: $\neg\neg x = x$.

Theorem. $CP \vdash P = Q \iff se(P) = se(Q)$

Proof. Easy (incl. *se*-equality is a congruence).

Note. *se*-equality is further called **Free valuation congruence** (FVC).

Evaluation trees for expressions with $\neg$, $\mathbin{\&\&}$, and $\mathbin{||}$:

$$se(\neg P) = se(P)\,[T \mapsto F, F \mapsto T] \qquad = se(F \triangleleft P \triangleright T)$$
$$se(P \mathbin{\&\&} Q) = se(P)\,[T \mapsto se(Q)] \qquad = se(Q \triangleleft P \triangleright F)$$
$$se(P \mathbin{||} Q) = se(P)\,[F \mapsto se(Q)] \qquad = se(T \triangleleft P \triangleright Q)$$

Example: for $a, b, c \in A$ we find

$$se(a \mathbin{\&\&} (b \mathbin{\&\&} c)) = se((a \mathbin{\&\&} b) \mathbin{\&\&} c) = ((T \trianglelefteq c \trianglerighteq F) \trianglelefteq b \trianglerighteq) \trianglelefteq a \trianglerighteq F$$

FVC-axioms (thus, valid wrt. se-equality) not mentioned before:

$$F = \neg T \qquad F \mathbin{\&\&} x = F$$
$$T \mathbin{\&\&} x = x \qquad x \mathbin{\&\&} F = \neg x \mathbin{\&\&} F$$
$$x \mathbin{\&\&} T = x \qquad (x \mathbin{\&\&} F) \mathbin{||} y = (x \mathbin{||} T) \mathbin{\&\&} y$$

$$(x \mathbin{\&\&} y) \mathbin{||} (z \mathbin{\&\&} F) = (x \mathbin{||} (z \mathbin{\&\&} F)) \mathbin{\&\&} (y \mathbin{||} (z \mathbin{\&\&} F))$$

Theorem (Staudt, 2012). "**Short-circuit logic** for Free VC"

For propositional formulae over A, T, F, $\neg$, $\&\&$, $||$, FVC is axiomatized by the seven axioms listed on the previous slide, and

$$\neg\neg x = x \qquad \text{(DNS)}$$

$$x \mathbin{||} y = \neg(\neg x \mathbin{\&\&} \neg y) \qquad \text{(def. of } ||\text{, implying DM's laws)}$$

$$(x \mathbin{\&\&} y) \mathbin{\&\&} z = x \mathbin{\&\&} (y \mathbin{\&\&} z) \qquad \text{(implying assoc. of } ||\text{)}$$

say E, thus $E \vdash P = Q \iff se(P) = se(Q)$.

Proof. Soundness (incl. congruence property) is easy.
Completeness is non-trivial (20^+ pages) and depends on:

- normal forms,
- decomposition properties of evaluation trees for $\&\&$ and $||$, and
- the existence of an inverse g of se for normal forms: $g(se(P)) = P$

3. Valuation Congruences

FVC (equationally axiomatized by CP)

$\subseteq$ Repetition-proof VC:
equationally axiomatized by $CP +$ two axiom schemes over A

$\subseteq$ Contractive VC:
equationally axiomatized by $CP +$ two axiom schemes over A

$\subseteq$ Memorizing VC: equationally axiomatized by $CP +$ one axiom
typical properties: $x \mathbin{\&\&} x = x$
$x \triangleleft y \triangleright z = (y \mathbin{\&\&} x) \mathbin{||} (\neg y \mathbin{\&\&} z)$

$\subseteq$ Static VC $\approx$ "sequential propositional logic":
equationally axiomatized by $CP +$ two axioms

These VC's are defined by varieties of *Valuation algebra's* [BP10].

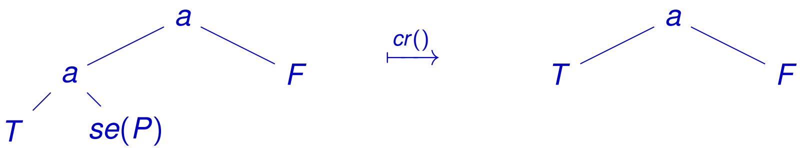
[BP15]: RpVC – MVC also have **simple semantics**: transformations on evaluation trees (cf. the use of truth tables in Propositional Logic).

Contractive VC: Subsequent occurrences of the same atom are contracted; equational axiomatization:

$$\begin{aligned} CP_{cr}(A) = CP \cup \ \{&(x \triangleleft a \triangleright y) \triangleleft a \triangleright z = x \triangleleft a \triangleright z, \\ &x \triangleleft a \triangleright (y \triangleleft a \triangleright z) = x \triangleleft a \triangleright z \mid a \in A\} \end{aligned}$$

Example: $a \mathbin{\&\&} (a \mathbin{||} x) = (T \triangleleft a \triangleright x) \triangleleft a \triangleright F = T \triangleleft a \triangleright F = a$

$se(a \mathbin{\&\&} (a \mathbin{||} P))$ and its contracted evaluation tree:



The transformation $cr : \mathcal{T}_A \to \mathcal{T}_A$ is the contraction function, and recursively traverses the tree.

A more concrete example for Contractive VC.

Programming with n Boolean registers. For $1 \leq i \leq n$ consider registers R_i with for $B \in \{\mathtt{T}, \mathtt{F}\}$,

- the atom $(\mathtt{set}{:}i{:}B)$ can have a side effect: it sets R_i to value B and evaluates in each state to `true`
- the atom $(\mathtt{eq}{:}i{:}B)$ has no side effect and evaluates to `true` if R_i has value B, and otherwise to `false`

Then all instances of $CP_{cr}(A)$ are valid, but not all instances of the stronger equation $x \mathbin{\&\&} x = x$ (valid under MVC): Let

$$t = (\mathtt{eq{:}1{:}F}) \mathbin{\&\&} (\mathtt{set{:}1{:}T})$$

and assume R_1 has initial value $\mathtt{F}$, then $\begin{cases} t & \text{evaluates to } \mathtt{true} \\ t \mathbin{\&\&} t & \text{evaluates to } \mathtt{false} \end{cases}$

Note. Not all valid eq's are derivable, e.g., $(\mathtt{eq{:}1{:}F}) \mathbin{\&\&} \neg(\mathtt{eq{:}1{:}F}) = F$.

Theorem [BP15, BP10]. $CP_{cr}(A) \vdash P = Q \iff cr(se(P)) = cr(se(Q))$

Corollary. "**Short-circuit logic** for Contractive VC"

For propositional formulae over $A, T, F, \neg, \mathbin{\&\&}, \mathbin{||}$,

$$\left\{\begin{array}{r} \neg x = F \triangleleft x \triangleright T, \\ x \mathbin{\&\&} y = y \triangleleft x \triangleright F, \\ x \mathbin{||} y = T \triangleleft x \triangleright y \end{array}\right\} \cup CP_{cr}(A) \vdash P = Q \iff cr(se(P)) = cr(se(Q))$$

Open question. Does a finite, equational axiomatization of CVC exist without the use of $.. \triangleleft .. \triangleright ..$?

(An approach as in [Staudt12] seems not possible.)

Note. Wrt. Repetition-proof VC we have a similar Theorem, Corollary, and open question.

4. Remarks and conclusions

4.1 Hoare's conditional (1985):

- Original approach: characterization of Propositional Logic
- Original definition: $P \triangleleft Q \triangleright R = (P \wedge Q) \vee (\neg Q \wedge R)$
  However, wrt. side effects the alternative, intuitive reading

  $$P \triangleleft Q \triangleright R \approx \textit{if } Q \textit{ then } P \textit{ else } R,$$

  is preferable: it suggests/prescribes a sequential, short-circuited interpretation

With this intuition AND the naturalness of *se*() AND the definitions of

$$\neg, \ \&\&\ , \ ||\ ,$$

it is evident that *CP* is most basic.

4.2 Sequential, propositional connectives:

T, $\neg$, and $\mathbin{\&\&}$ (and/or definable counterparts) seem primitive:

- For example, strict (complete) sequential evaluation of conjunction, notation $\mathbin{\&}$, is defined by

$$x \mathbin{\&} y = (x \mathbin{||} (y \mathbin{\&\&} F)) \mathbin{\&\&} y$$

  (one more argument to include T (and F) in this setting)

- BUT, a sequential version of XOR, notation $\oplus$, is defined by

$$x \oplus y = \neg y \triangleleft x \triangleright y$$

  and cannot be defined modulo Free, Repetition-proof, or Contractive VC with T, $\neg$, and $\mathbin{\&\&}$ only

Hence: $.. \triangleleft .. \triangleright ..$ is a convenient primitive, and the possible side effects of the atoms of interest determine an appropriate VC.

4.3 Transformations on Evaluation trees for more identifying VC's:

- Transformation to a Repetition-proof evaluation tree is **natural** and **simple** (cf. [ERO60]); semantics by term rewriting is not easy in this case, e.g. $(x \triangleleft a \triangleright F) \triangleleft a \triangleright F \ \rightarrow \ (x \triangleleft a \triangleright x) \triangleleft a \triangleright F$
- Transformation to a Contractive or Memorizing evaluation tree is also **N&S** (see [BP15])
- Transformation to a static evaluation tree is more complicated and requires an ordering of the atoms [Hoare85 + BP15]

4.4 Extensions of $.. \triangleleft .. \triangleright ..$ to many-valued logic's are easily defined (and seq. evaluation often provides good intuitions):

E.g., Belnap's 4VL [PZ07], or 5VL [BP99] = Belnap's 4VL + Bochvar's constant M which majorizes all truth values